UNIVERSITY OF ALBERTA

BLOCKING ANALYSIS AND DESIGN OPTIMIZATION OF A CLASS OF CIRCUIT-SWITCHED MULTICAST NETWORKS

By

ASHISH DUGGAL

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1996

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-18254-1

Canada

UNIVERSITY OF ALBERTA

Library Release Form

Name of Author: ASHISH DUGGAL

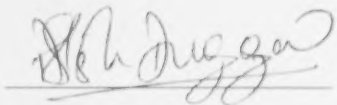
Title of Thesis: Blocking Analysis and Design Optimization of a Class of Circuit-Switched Multicast Networks

Degree: MASTER OF SCIENCE

Year this Degree Granted: 1996

Permission is hereby granted to the University of Alberta to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly, or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion maybe printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Ashish Duggal

10924-16 Avenue

Edmonton, Alberta

T6J 5G6

Date: 31 JULY 96

UNIVERSITY OF ALBERTA

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled

"Blocking Analysis and Design Optimization of a Class of Circuit-Switched Multi-cast Networks"

submitted by **Ashish Duggal** in partial fulfilment of the requirements for the degree of **Master of Science**.

Wayne D. Grover (Supervisor)

Mike H. MacGregor (Co-Supervisor)

Anup Basu (External)

Bruce Cockburn (Internal)

Date: July 30/96

DEDICATED TO

My Parents: Shiv and Neena Duggal

ABSTRACT

We present an analytical model, a discrete event emulator, and a design optimization method for a type of wide-area multicast video-conferencing service which interconnects subscriber locations via circuit-switched DS-1 multicast control units (MCUs) arranged to form a switching network. The analytical model can be used to predict the blocking of a given topology under a specified load. Optimization leads to a configuration of MCUs in which end-to-end blocking meets a target specification with a minimum total number of MCUs.

Modelling the end-to-end blocking of this class of network requires aspects of conventional traffic theory for full availability group blocking and aspects of multistage link-trunking network analysis via channel graphs, extended for multicast considerations. The theory is validated by a network emulator which is designed to be a functional model of call-by-call handling in an actual multicast environment. The optimization method is based on finding the split in MCU port allocations for which the sum of access site blocking and inter-MCU junctor blocking is minimized.

The analytical model and the accompanying multicast network emulator are suitable for use in a planning system to support operation of the intended multicast services. The analytical model has been useful in providing insights about the fundamental factors influencing the performance of such networks but it cannot as easily deal with effects such as heterogeneous traffic flows from different user groups and asymmetry in the network linking patterns. The network emulator has no such limitations and is therefore also useful in the practical design of networks for specific customer circumstances

ACKNOWLEDGEMENTS

First of all, I would like to thank **Dr. Wayne D. Grover**, my supervisor, for having confidence in me and giving me the opportunity to work with him. His help and guidance throughout my work are deeply appreciated. I am also very thankful to **Dr. Mike MacGregor** (Co-Supervisor) for his effort and timely guidance in both technical and personal senses. Mike also introduced me to the wonderful world of Kayaking :).

During my graduate work at TRLabs, I have come across a lot of friends whose friendship I will always cherish. The members of **Networks and Systems Mafia: Demetrios**, **Rainer**, **Vipul**, **Danny**, **Jim** and **Jason;** and **Ping** and **Prasad** from our rival Photonics group. Demetrios had to bear working with me in the same computer cluster, but I think he enjoyed my constant bickering at him. In return for my torturing him everyday, Demetrios helped me out with the Mapping Algorithm for calculating the state probability of network links. Special thanks to Rainer for giving me kayaking lessons (which nearly drowned me! just kidding). Jim was very helpful in giving me tips about which car to buy. Jim, I can't afford the BMW yet!

I will always be thankful to **Mamaji** and **Mamiji** for giving me a home away from home and **Chacha** and **Chachi** for having confidence in me.

I would like to thank my Parents and Grandparents for being supportive while being thousands of miles away. I could not have done it without their love.

I am also grateful to the members of my committee: Dr.'s Grover, MacGregor, **Basu** and **Cockburn** for their helpful comments and suggestions. Jim Sandham of AGT Ltd. is thanked for introducing us to the problem of multicast networks. Dr. Grover, TRLabs and University of Alberta are thanked for their financial support.

TABLE OF CONTENTS

1.0 INTRODUCTION 1

1.1 PROBLEM INTRODUCTION 1

1.2 OVERVIEW & OBJECTIVES 1

1.3 PRIOR WORK AND ISSUES 3

1.3.1 SWITCHING NETWORKS 3

1.3.2 MULTICAST TRAFFIC AND BLOCKING 4

1.3.3 MULTICAST ROUTING 5

1.4 THESIS OUTLINE 5

2.0 TRAFFIC & SWITCHING THEORY BACKGROUND 7

2.1 TRAFFIC ENGINEERING 7

2.1.1 TRAFFIC CHARACTERIZATION 7

2.2 GRADE OF SERVICE & SERVER GROUPS 9

2.3 BLOCKING FORMULAS 11

2.3.1 ERLANG B THEORY 11

2.3.2 ENGSET TRAFFIC THEORY 12

2.4 SWITCHING NETWORKS 13

2.4.1 DOUBLE-SIDED SWITCHES 13

2.4.2 ONE-SIDED SWITCHES 14

2.4.3 NON-BLOCKING & FAN-OUT SWITCHES 14

2.4.4 DOUBLE-SIDED MULTISTAGE NETWORKS 15

2.4.5 ONE-SIDED MULTISTAGE NETWORKS 16

2.4.6 BLOCKING IN MULTISTAGE NETWORKS 18

2.4.7 NON-BLOCKING SWITCHING NETWORKS 19

2.4.8 CHANNEL GRAPHS 19

2.5 SUMMARY 20

3.0 NETWORK MODEL 21

3.1 NETWORK TOPOLOGY 21
3.1.1 ACCESS NETWORK 21
3.1.2 INTER-MCU SWITCHING NETWORK 22
3.2 MULTICAST CALLING & BLOCKING DEFINITIONS 24
3.3 MULTICAST CALL ROUTING MODEL 25
3.4 SUMMARY 26

4.0 ANALYTICAL MODEL 28

4.1 BLOCKING CHARACTERISTICS 28
4.2 STRUCTURE OF THE ANALYTICAL MODEL 29
4.3 OFFERED TRAFFIC 31
4.4 SITE CONGESTION, P_s 32
4.5 ACCESS LINK/PORT UTILIZATION, a_c 33
4.6 DEFLECTION PROBABILITY, P_{def} 34
4.7 NETWORK LINK UTILIZATION, a_n 36
4.8 NETWORK BLOCKING 38
4.8.1 NETWORK BLOCKING FOR POINT-TO-POINT TRAFFIC 38
4.8.2 MULTICAST NETWORK BLOCKING 40
4.8.2.1 PROBABILITY OF NETWORK LINKS BEING FREE, $P_f(k)$ 42
4.8.2.2 BLOCKING CORRECTION IN MAPPING EQUATIONS 47
4.8.2.3 PROBABILITY OF NETWORK BLOCKED CONNECTION ATTEMPT 50
4.8.3 SUMMARY OF NETWORK BLOCKING MODEL, ... 51
4.9 COMPUTATIONAL METHODOLOGY 51
4.10 SUMMARY 53

5.0 MULTICAST NETWORK EMULATOR 54

5.1 EMULATOR INPUTS 54
5.2 CALL GENERATION 55
5.3 MULTICAST CALL ARRIVAL, PROCESSING AND DEPARTURE TIMES 56
5.3.1 NEXT CALL ARRIVAL 56
5.3.2 CALL PROCESSING 57
5.3.2.1 ROUTING ALGORITHM 58
5.3.3 CALL DEPARTURE SCHEDULING 58
5.3.4 CALL DEPARTURE EVENT 59
5.4 EMULATOR OUTPUTS 59
5.4.1 TRANSIENT EFFECTS 59
5.4.2 SITE BLOCKING 60
5.4.3 ACCESS LINK UTILIZATION, a_c 61
5.4.4 DEFLECTION PROBABILITY, P_{def} 62
5.4.5 NETWORK LINK UTILIZATION, a_n 62
5.4.6 NETWORK BLOCKING, P_n 63
5.5 TESTING & VALIDATION OF THE NETWORK EMULATOR 63
5.6 SUMMARY 65
6.0 TEST RESULTS 66
6.1 TEST NETWORKS 66
6.1.1 NETWORK A 66
6.1.2 NETWORK B 70
6.1.3 NETWORK C 70
6.2 TESTING PROCEDURE 70
6.3 START-UP TRANSIENT 71
6.4 EXPERIMENTAL VALIDATION OF EQUIVALENCE OF TIME CONGESTION and CALL CONGESTION IN THE ACCESS GROUPS 72

6.5 SITE CONGESTION/BLOCKING, P_s 75
6.6 ACCESS PORT UTLIZATION, a_c 77
6.7 PROBABILITY OF DEFLECTION, P_{def} 80
6.8 NETWORK LINK UTILIZATION, a_n 82
6.9 NETWORK BLOCKING, P_n 85
6.10 END-TO-END BLOCKING, P_{end} 88
6.11 SUMMARY 90

7.0 MULTICAST NETWORK DESIGN OPTIMIZATION 91

7.1 MOTIVATION 91
7.2 OPTIMIZATION STRATEGY 93
7.2.1 OPTIMIZATION PROCEDURE 95
7.3 TESTS & RESULTS 96
7.4 SUMMARY 97

8.0 VARIANTS ON THE NETWORK MODEL 98

8.1 OPTIMIZED MULTICAST ROUTING ALGORITHM IN EMULATOR 98
8.2 MULTIPLE INTER-STAGE LINKS 102
8.3 COMPLETE BLOCKING RATHER THAN PARTIAL CALL SETUP 104
8.4 SUMMARY 105

9.0 SUMMARY, CONCLUSIONS AND FUTURE WORK 107

9.1 THESIS REVIEW 107
9.2 CONCLUDING DISCUSSION 108
9.3 FUTURE WORK 109

BIBLIOGRAHY 110

APPENDIX A Algorithm for calculating the state probabilities of network links 113

LIST OF FIGURES

Figure	Description	Page
2.1	Full availability server groups	10
2.2	Double-sided switching module	13
2.3	One-sided switching module	14
2.4	A fanout switch	15
2.5	Three-stage double-sided switching network	16
2.6	Two-stage single-sided switching network	17
2.7	Channel graph of a three stage network	20
3.1	Network topology	22
3.2	Multicast routing model including deflection	25
4.1	End-to-end blocking model	29
4.2	Functional Interrelationships of the Analytical Model	30
4.3	Access port utilization	33
4.4	Deflected connection	34
4.5	Multicast traffic amplification at first stage MCU	36
4.6	Channel graph for point-to-point traffic	39
4.7	Multicast channel graph for a multicast call request	41
4.8	Markov Chain for network link states (d=3, m=6)	43
4.9	Channel graph for one source-destination pair	50
5.1	Sample Network File	54
6.1	Network A	67
6.2	Network B	68
6.3	Network C	69
6.4	Start-up transient in Network A for D=3	72
6.5(a)	Time congestion vs. Call congestion for Network A	73
6.5(b)	Time congestion vs. Call congestion for Network B	74
6.5(c)	Call congestion vs. Time congestion for Network C	74
6.6(a)	Site blocking in Network A	75

6.6(b) Site blocking in Network B 76
6.6(c) Site blocking in Network C 76
6.7(a) Access port utilization for Network A 77
6.7(b) Access port utilization for Network B 78
6.7(c) Access port utilization for Network C 78
6.8 Binomial vs. Emulator: State probabilities of access links 79
6.9(a) Deflection probability in Network A 81
6.9(b) Deflection probability in Network B 81
6.9(c) Deflection probability in Network C 82
6.10(a) Network link utilization in Network A 83
6.10(b) Network link utilization in Network B 83
6.10(c) Network link utilization in Network C 84
6.11 State probabilities of network links 85
6.12(a) Network blocking in Network A 86
6.12(b) Network blocking in Network B 87
6.12(c) Network blocking in Network C 87
6.13(a) End-to-end blocking in Network A 88
6.13(b) End-to-end blocking in Network B 89
6.13(c) End-to-end blocking in Network C 89
7.1 Site blocking vs. Network Blocking for varying n 92
7.2 Partitioning of first stage MCU 93
7.3 Network Topology 94
7.4 Optimization curves: *N=20, D=3, Ao=0.1 erl, z=2, C=16* 97
8.1 Greedy routing example 98
8.2 Optimal routing example 99
8.3 Site blocking in Network A: greedy vs. optimal routing 100
8.4 Network blocking in Network A: greedy vs. optimal routing 101
8.5 Network blocking in Network B: multiple inter-stage links 103
8.6 End-to-end blocking in Network A: soft blocking vs. hard blocking 105

LIST OF SYMBOLS

A	traffic intensity
A_l	offered traffic lost due to blocking
A_c	amount of traffic carried
A_o	traffic intensity originated by a site
A_{tot}	offered terminating traffic at a site
a_c	access link utilization including deflection probability
a_c'	access link utilization excluding deflection
a_n	network link utilization
C	total number of ports per MCU
D	maximum number of destinations allowed in a call
d	number of destinations in a multicast call
h	holding time
m	number of second stage MCUs and the number of network links on each first stage MCU
N	number of user sites in the multicast network
n	number of links on each first stage MCU for user access
P_s	site blocking at the access ports
P_n	blocking in the two-stage switching network
P_{def}	probability of a deflected connection
$P(d)$	distribution of number of destinations per call
p	number of access ports per site
V	traffic volume
X	number of first stage MCUs in the network
z	number of sites having access ports on the same first stage MCU
γ	arrival rate

1.0 INTRODUCTION

1.1 PROBLEM INTRODUCTION

Traditionally, circuit switched networks have been used primarily for communicative traffic such as telephony. This traffic is characterized by point-to-point connections (one subscriber needs to talk to only one other subscriber at a time) and by moderate bandwidth requirements such as 64 kbit/s. In recent years, there has been an increase in the number of applications requiring group-based communications where more than two subscribers wish to communicate among themselves. This has led to the need for multicast networks. *Multicasting* is the simultaneous exchange of information between more than two users in the network. In contrast, *broadcasting* is the transmission of a message to all users in the network, regardless of whether any one of them wishes to receive the message. Thus, multicasting is the selective broadcast of a message from one source to a specific subset of possible destinations. Multicast applications include teleconferencing, video-conferencing, pay-per view CATV programs and interactive distance learning.

More recently, a particular multicast video-conferencing service has been under study to provide wide-area educational and business networks via circuit switching elements called *multipoint control units* (MCUs) distributed through the network. MCUs [1], [2], [3] are small switching devices that are intended to be used in a network designed to carry multicast videoconferencing traffic. The MCU technology is relatively new (mid 1991) and allows the use of services for a range of bit rates ranging from 56 kbps to DS-1 (1.544 Mbps) and E1 (2.048 Mbps) rates. The hardware for multicast bridging and switching in the MCUs is relatively well understood, but it has been unclear how to optimally provision and configure the MCUs in a network so that the end-to-end blocking of random arrival multicast traffic meets a target level, while using a minimum number of multicast control units.

1.2 OVERVIEW & OBJECTIVES

Multicast videoconferencing is emerging as an efficient and cost effective way to communicate. It has applications in education, the medical community and business as well. A

multicast video-conference can save time as well as money, when it displaces high travel costs.

Circuit-switched networks have a single dominant performance metric: the end-to-end blocking probability of a call request. The blocking in a network is related to the amount of traffic offered and the configuration of the network resources. The *grade of service* (GOS) is usually specified as a target level of blocking in the design busy hour and dictates the provisioning of the network resources to meet the blocking objective at a given traffic level. During all but the design level peak times, actual blocking probabilities are far lower than the busy hour GOS targets. The work presented is based on the assumption of statistical equilibrium and does not consider time varying traffic. The motivating factor is videoconferencing but the rate at which the network operates has no affect on the blocking in the class of networks we propose. Thus, given a test network, we need to be able to evaluate its performance under a given set of traffic conditions. Even though the work presented considers only circuit-switched applications, in future the network might be able to handle both packet and circuit-switching.

The objective of this thesis is to establish a method for optimizing the design and evaluating the performance of a class of circuit-switched multicast networks. The networks considered provide wide-area multicast videoconferencing by interconnecting private studios located on various user premises. We outline a model of the network under consideration and formulate a number of assumptions which allow us to create hypothetical test networks of this type with varying dimensions but having similar properties. The work has followed two main approaches and led to these main areas of contribution:

1) Analytical Modelling: An analytical model has been developed to calculate the end-to-end blocking probability in a given multicast network. The model relates the blocking in the network to the multicast traffic and the configuration of network resources. The end-to-end blocking involves aspects of ordinary Erlang traffic theory for full availability groups and aspects of multistage link-trunking network analysis via multicast channel graphs.

2) Multicast Network Emulator: The emulator is a functional model of call-by-call handling in the multicast network and is used to validate the analytical model. It is also useable for performance modelling of detailed network designs for specific customer circumstances.

3) Network Design Optimization: The thesis also describes the theory that shows how to optimize the configuration of the network using a minimum number of MCUs while guaranteeing a particular grade of service. Both the analytical model and the emulator are intended for use in a planning system to support operation of the planned multicast services.

1.3 PRIOR WORK AND ISSUES

1.3.1 SWITCHING NETWORKS

The design of switching networks has been under study since the deployment of the first telecommunication network. Traditional telephony networks were designed to carry point-to-point traffic with a small probability of blocking. As the user demand increased, there was a need to provide better service; the design of non-blocking switching networks was introduced by Clos [4] in 1953. He described a method of designing switching networks in which it is always possible to establish a connection from an idle input to an idle output regardless of the number of calls existing in the system. Such switching networks are referred to as "*Clos Networks*" and laid the foundation for future work in switching network design. One drawback of the non-blocking networks proposed by Clos was the relatively large number of switching elements required to guarantee point-to-point service. The idea of rearrangeable networks requiring fewer elements were introduced in which it is always possible to connect an idle input-output pair by re-routing or rearranging existing connections[5], [6]. With advances in technology, the analysis was extended to include multicast connections, including results for non-blocking and rearrangeable multiconnection networks [7], [8], [9], [10], [11]. To achieve this multiconnection, the switches are assumed to have fanout capability in which, an input port on a switch can connect to all output ports.

Multicasting requires more network resources on a call basis, so the number of switching elements required in a non-blocking multicast network is greater than for point-to-point non-blocking networks. Therefore, from a service provider's perspective there is a need to provide quality service at the minimum possible capital cost. This thesis addresses this issue through optimization of economic, low blocking multicast switching networks with fewer switching modules than fully non-blocking networks.

1.3.2 MULTICAST TRAFFIC AND BLOCKING

Relatively little can be found in the literature on performance analysis and optimization of blocking multicast switching networks. The issue of internal link blocking in point-to-point multistage switching networks is well understood. Takagi introduced the notion of a *channel graph* [12], which is a simpler representation of a multistage network and is used in blocking analysis. However, the idea of channel graphs was originally intended only for networks carrying point-to-point traffic. We are unaware of any published work where the idea of channel graphs is used to calculate blocking in a multicast network.

The issue of blocking in trunk groups for multicast traffic is also well understood. It involves the use of basic Erlang theory for full availability server groups [12], [13]. However, most of the work in the area of blocking in multicast switching networks makes no attempt towards network optimization. Wang et al. [14] and Labetoulle [15] treat trunk congestion in a circuit switched network with 2 types of traffic with different mean holding times. Liao and Roberts [16] propose a model of the videoconferencing reservation traffic process in which the essential parameters are the size and distribution of the user sites and the mean busy hour rate. They assume that most of the calls will be point-to-point and do not attempt to relate the traffic model to the grade of service. The analysis they present is not validated by simulation or experiment. An attempt is made to dimension trunk groups to a given level of blocking, but is restricted to point-to-point traffic [17]. More recently, Zegura evaluates blocking probability in multistage networks and allows for multicast connection [18]. But the analysis is considerably different than our approach and in addition, no attempt is made to optimize the multicast network topology.

1.3.3 MULTICAST ROUTING

The routing algorithm used to set up connections is important in blocking networks. An efficient routing algorithm should make optimal use of the network resources while satisfying connection requests. Chow has designed and implemented three multicast path finding algorithms [19] with a performance orientation on the time taken by each algorithm rather than on the blocking levels in the network based on traffic offered. Leung and Yum have formulated the *multiple destination routing (MDR) problem* as an integer programming problem [20]. Their emphasis is on reducing the time complexity of the computations rather than doing a blocking analysis based on the routing. No specific traffic assumptions are made nor any attempt to optimize the network in [20]. Harrison researched the performance of multicast routing methods based on multicast traffic [21]. Here the emphasis is on simulation and there is no validation of simulation results obtained analytically. Also, the network topology considered is a regularly connected mesh which is economically unattractive and unrealistic. The author does not attempt to design a practical network for a specific multicast service based on given traffic levels and grade of service objectives. Lee et al. investigate representative multipoint routing algorithms from a number of viewpoints [22]: computational complexity, dynamic re-configuration and bandwidth efficiency. But they do not consider performance aspects arising from the routing of multicast traffic.

1.4 THESIS OUTLINE

In this thesis, we address some issues that have not previously been dealt with. We focus on the design of a class of multicast networks which will provide video-conferencing service to subscribers over a wide-area. The thesis is organized as follows:

Chapter 2 provides background on traffic theory, grade of service and circuit-switching networks.

Chapter 3 presents the network model for the particular type of multicast network which we consider. In particular, it outlines the assumptions made about the network topology, multicast routing model and the multicast traffic offered to the network.

Chapter 4 is devoted to analysis of the blocking performance of multicast networks of the type in Chapter 3. Chapter 4 also outlines the procedure for obtaining numerical results using the interrelated system of equations that is developed.

Chapter 5 describes the multicast network emulator. This is a functional model of a multicast network with call-by-call simulation. The emulator is used as a benchmark to validate the analytical model presented in Chapter 4 by comparing performance results obtained for a given test network.

Chapter 6 reports testing of the analytical model against the emulator results for various test networks.

Chapter 7 describes an optimization procedure based on the analytical model which minimizes network resources while satisfying the grade of service objective.

Chapter 8 investigates some effects of variations in the network model on the performance of the test networks. In particular, we look at variations in the linking pattern in the switching network and changes to the routing algorithm.

Chapter 9 summarizes the work while identifying the major research findings and outlining areas of further research.

2.0 TRAFFIC & SWITCHING THEORY BACKGROUND

Telecommunications networks for public and private use are among the largest interconnected systems in the world. The purpose of this chapter is to familiarize the reader with some key concepts underlying the design of circuit switched networks. These concepts are relevant towards understanding the research work described in this thesis.

2.1 TRAFFIC ENGINEERING

There are over 350 million public network telephone terminals in addition to public and private networks for data, video, finance, etc. [12]. Within these networks, all terminals must be fully interconnectable with each other. In designing a practical network, network planners achieve cost savings by providing sufficient transmission resources such that the probability of the network failing to provide immediate service is acceptable. The principle behind sharing of network resources is that user demands occur randomly with low utilization and occur at random times. It is the discipline of traffic theory that mediates the trade-off between the amount of equipment and the resulting quality or grade of service.

2.1.1 TRAFFIC CHARACTERIZATION

The following definitions are used to model telecommunications traffic flow in a network [23]:

(a) Arrival rate (γ): This is defined as the average number of calls originating per unit time from one or a group of terminals or users in the network. Calls from different users originate randomly in time. If α is the number of calls initiated by N users or sources in a period T, the arrival rate of the group is:

$$\gamma = \frac{\alpha}{T} \qquad \text{(Eq. 1)}$$

Due to the random nature of a call arrival, the arrival rate may be restated as the probability of a call request arising within a short interval of time δt. The inverse of the arrival rate is the inter-arrival period.

(b) Holding time (h): The mean time for which all calls last is called its holding time. From theory and measurement, it is found that the probability of a call lasting t seconds or more is given by the negative exponential distribution:

$$P(T \geq t) = e^{-t/h} \quad \text{(Eq. 2)}$$

The probability per unit time of a call ending is constant and therefore, independent of the length of time a call has already been in progress. This implies that call departure from occupied trunks is a memoryless process. In general, there is an important duality between the Poisson and negative exponential distribution that has two useful analytical implications:

- Call arrivals which are Poisson distributed have call interarrival periods which follow a negative exponential distribution.
- Call holding times which are exponentially distributed give rise to call departures which follow a Poisson process.

(c) Traffic volume (V): If α is the number of call requests originating from a group of sources in a period T, and h is the mean holding time, the traffic volume is defined as:

$$V = \alpha \cdot h \quad \text{(Eq. 3)}$$

The unit of traffic volume in North America is "ccs" which stands for "hundred call seconds". 1 ccs is a volume of traffic equal to one circuit being busy for 100 seconds or 10 circuits being busy for 10 seconds, etc.

(d) Traffic intensity (A): Traffic intensity is the traffic volume divided by the time T over which this traffic occurs:

$$A = \frac{\alpha h}{T} = \frac{\gamma}{\mu} \text{ where } \mu = 1/h \quad \text{(Eq. 4)}$$

Traffic intensity is dimensionless, but is given units of erlangs, denoted E or erl. Although dimensionless, the erlang can be viewed as:

- the flow rate of traffic per unit time in units of ccs/hr.

or

- the ratio of the actual traffic volume per unit time to the traffic represented by one circuit that is continually busy.

Using the definition of ccs/hr, we can conclude that 1 ccs/hr = 100 seconds/3600 seconds = $0.02\bar{7}$ E. Numerically, for a group of circuits or terminals, the traffic intensity A is equivalent to:

- The average number of circuits simultaneously busy in the group.
- The expected number of call arrivals per unit holding time.
- The number of circuits required to completely carry the offered traffic if each circuit were operating at 100% occupancy.

For example, 2 erlangs is produced by 2 terminals busy 100% of the time or 8 terminals with 25% average occupancy or utilization.

2.2 GRADE OF SERVICE & SERVER GROUPS

The traffic offered by a group of terminals to a network is shared between a group of trunks or junctors called *server group* (Figure 2.1). Each server group may carry traffic from one switching centre to another. If an inlet can connect to any trunk in the group, the server group is called a *full availability group*.

Resource sharing implies that on occasion a request for service will not be served immediately due to the unavailability of a free server. This denial of service is called *blocking*, and in most practical networks low levels of blocking are acceptable. The amount of traffic originating from a group of sources and offered to a server group is called the *offered traffic*, A_o. Due to blocking, some of the offered traffic will be lost, A_l, or not carried by the servers in the group. The amount of traffic actually serviced and carried by trunk groups is called *carried traffic*, A_c, as shown in Figure 2.1.

These quantities are related:

$$A_o = A_c + A_l \qquad \text{(Eq. 5)}$$

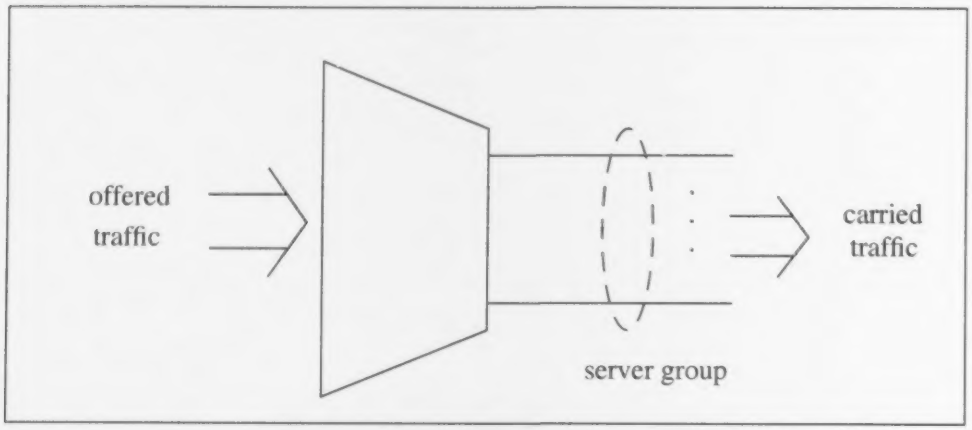


Figure 2.1. Full availability server groups

The amount of traffic lost is dependant on the average blocking:

$$A_l = P(B) \cdot A_o \qquad \text{(Eq. 6)}$$

Grade of service or the *probability of blocking*, *P(B)*, is the fraction of the total traffic lost:

$$P(B) = \frac{A_o - A_c}{A_o} = \frac{A_l}{A_o} \qquad \text{(Eq. 7)}$$

It is the congestion of servers that causes blocking. For example, consider a system with 2 sources and 1 server. Congestion of servers can be specified in two ways:

(a) *Time congestion.* This is the fraction of the observed time over which all servers are busy. For example, if the system is observed for 10 minutes and the system is in the "all servers busy" state for a total of 2 minutes, then the time congestion is 0.2.

(b) *Call congestion.* The proportion of call attempts that do not find a free server immediately is call congestion. In the above example, if 20 calls arrive in 10 minutes and 2 of them do not find a free server, then call congestion is 0.1. Note that the 10 minute duration is not relevant because we are looking for the fraction of calls rejected.

There can be a difference between time and call congestion because, although all servers may be busy, there may be a reduced probability of a next call arrival when source population finiteness is a significant effect.

2.3 BLOCKING FORMULAS

This section presents statistical analyses that relate blocking, traffic and number of servers, whilst assuming statistical equilibrium. *Statistical equilibrium* implies that the probability per unit time of a new call request is equal to the probability of a call termination. Hence, on average, the number of connections in progress remains constant. The Erlang B and Engset traffic theories are two of the formulas in telecommunications theory that relate four quantities: blocking probability, offered traffic, the number of sources and the number of servers. Both these theories assume that the arrival rate is a memoryless Poisson process and that holding times are exponentially distributed. It is also assumed that each source can connect to any free server.

2.3.1 ERLANG B THEORY

The *Erlang B* theory is valid under the following assumptions:

(a) The source pool is infinite relative to the number of servers.

(b) Blocked calls are discarded with zero holding time and do not reappear during the period of observation. This is called the *blocked calls cleared* or BCC assumption.

If A is the total traffic offered to N servers, then the probability of blocking with infinite sources and the BCC assumption is [12]:

$$B(N,A) = \frac{A^N/N!}{\sum_{i=0}^{N} A^i/i!} \qquad \text{(Eq. 8)}$$

Usually, network planners fix the blocking objective and find the number of servers required to carry the offered traffic at the desired GOS. Since the source pool is infinite compared to the number of servers, the probability of a new call is independent of the number already in progress and therefore call congestion is equal to time congestion for Erlang B systems.

2.3.2 ENGSET TRAFFIC THEORY

Erlang B theory does not consider that the arrival rate strictly decreases when one of the sources enters service. But in some traffic engineering situations it is necessary to consider the effects of a finite number of sources. *Engset theory* defines the probability of blocking with a finite number of sources M and blocked calls cleared as [12]:

$$E(M,N,A) = \frac{\left(\frac{A}{M-A}\right)^N \cdot \binom{M}{N}}{\sum_{i=0}^{N} \left(\frac{A}{M-A}\right)^i \cdot \binom{M}{i}} \qquad \text{(Eq. 9)}$$

Note that as $M \to \infty$ Engset becomes Erlang B. In practise $M < 10 \cdot N$ is a common rule of thumb for needing to consider finite source effects. The Engset formula above calculates the time congestion. But due to the limited number of sources, the probability of a call arising will depend on the number of sources all ready being served. Therefore, an alternate Engset formula for call congestion also exists [12]:

$$E(M,N,A) = \frac{\left(\frac{A}{M-A}\right)^N \cdot \binom{M-1}{N}}{\sum_{i=0}^{N} \left(\frac{A}{M-A}\right)^i \cdot \binom{M-1}{i}} \qquad \text{(Eq. 10)}$$

When our analytical model was first developed we thought that Engset theory would be applicable. But all the analysis and results presented in this thesis are based on the Erlang B formula. Engset may be useful in certain scenarios which are documented as future work. Other traffic theories are valid under different scenarios. For example, Erlang C calculates the blocking for a system under the blocked calls wait (BCW) and infinite sources assumption.

2.4 SWITCHING NETWORKS

Switching permits the network to be built economically by the concentration of transmission facilities. These facilities are the trunks connecting switching nodes. Each connection has an input and one or more output(s) depending on whether the call is point-to-point or multicast.

2.4.1 DOUBLE-SIDED SWITCHES

These switches have distinct sets of input and output ports. A switch with n inputs and m outputs is referred to as an $(n \times m)$ switch and is illustrated in Figure 2.2.

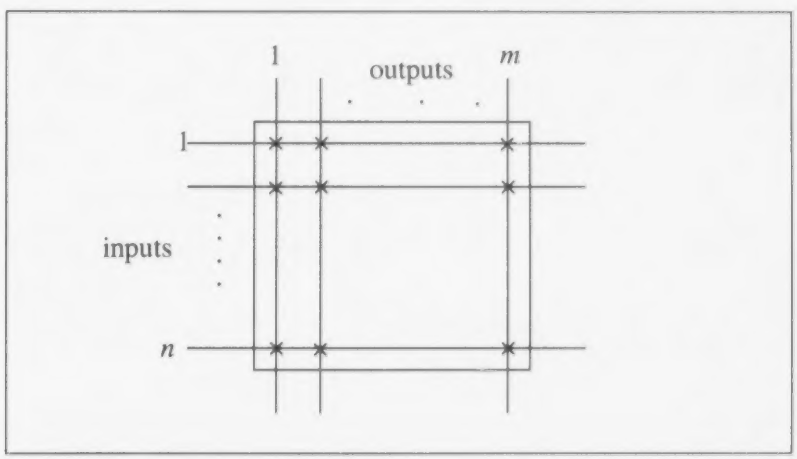


Figure 2.2. Double-sided switching module

An (n x m) switch uses $(n \cdot m)$ crosspoints, each shown as an x, where each crosspoint represents a possible connection between an input-output pair. In a square switching mod-

ule n=m. A double-sided switch assumes a unidirectional flow of traffic through its ports. In such an implementation traffic can arrive only on an input port and leave only on an output port.

2.4.2 ONE-SIDED SWITCHES

Unlike double-sided switches, *one-sided switches* have only one set of ports that act as both inputs and outputs (see Figure 2.3). Such switches assume a bi-directional flow of traffic with any of the ports in the switch being capable of sourcing or sinking traffic through the switch matrix. The port that originates traffic to the switch is called the input port and the port that sinks the traffic is called the output port. This reduces the number of crosspoints required to $n \cdot (n-1)/2$. A one-sided switch is also called a *folded matrix network* and is usually drawn as a right-angled triangle with ports on the perpendicular to the base.

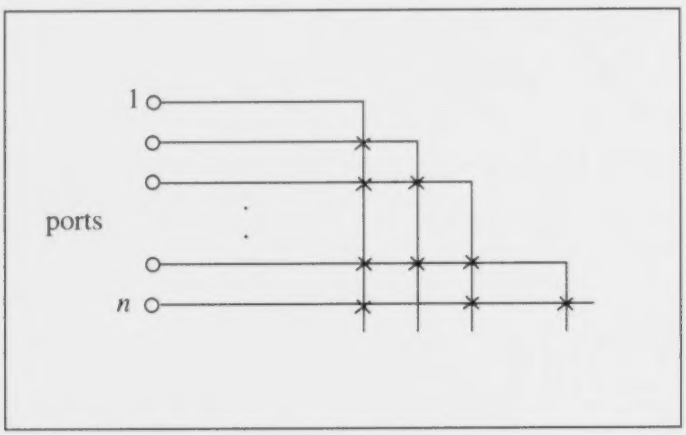


Figure 2.3. One-sided switching module

2.4.3 NON-BLOCKING & FAN-OUT SWITCHES

Double-sided and one-sided switches that provide for a connection between any pair of free ports regardless of other connections they may be carrying are referred to as internally *non-blocking switches*. For a point-to-point call, it is only necessary for a switch to make a connection between one pair of ports. But for certain applications involving multicast traf-

fic, an input port may be required to connect to more than one output port. Switches that support this technology are called *fanout* or *multicast switches*, as depicted in Figure 2.4.

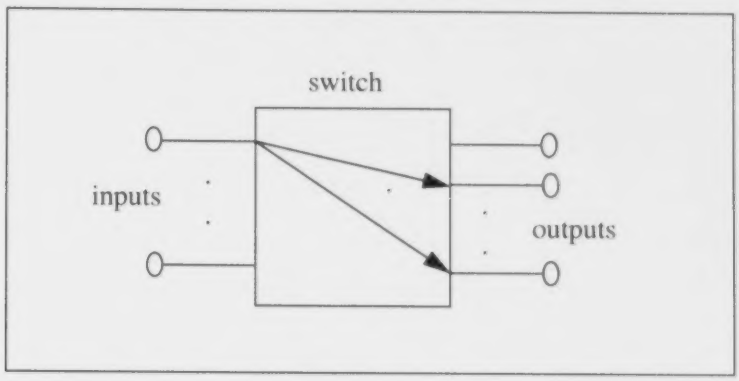


Figure 2.4. A fanout switch

2.4.4 DOUBLE-SIDED MULTISTAGE NETWORKS

Multistage switching networks are composed of switching elements distributed in more than one stage. In an $(N \times M)$ *double-sided multistage network* with N input ports and M output ports, the switching modules used as the building blocks to implement the network are double-sided. The inputs of the first stage switch modules are referred to as the *network input ports*. The first stage is often called the *input stage* of the network. The output ports of the last stage switches are referred to as the *network output ports* and the last stage is referred to as the *output stage*. The switching modules in different stages are interconnected by means of *links*, which are dedicated, fixed connection paths between the output ports of switches in stage i and input ports of switches in stage $i+1$. Switches in the same stage are not interconnected among themselves.

In general, an $(N \times M)$ three stage network [9] has r_1 $(n_1 \times m)$ switch modules in stage 1, m $(r_1 \times r_2)$ switches in the middle stage and r_2 $(m \times n_2)$ switches in stage 3. A general schematic of an $(N \times M)$ three-stage network is shown in Figure 2.5. There is one link between every pair of the first and second stage switches and between every pair of second and third stage switches. The flow of traffic through the network is unidirectional.

The traffic originates from the N input ports, travels through the three stages serially and flows out of the M output ports. A sample connection is shown in Figure 2.5. Two unidirectional connections are made in the network to establish one bi-directional connection. The links connecting the middle stage switch to the first and the third stage switches must be free.

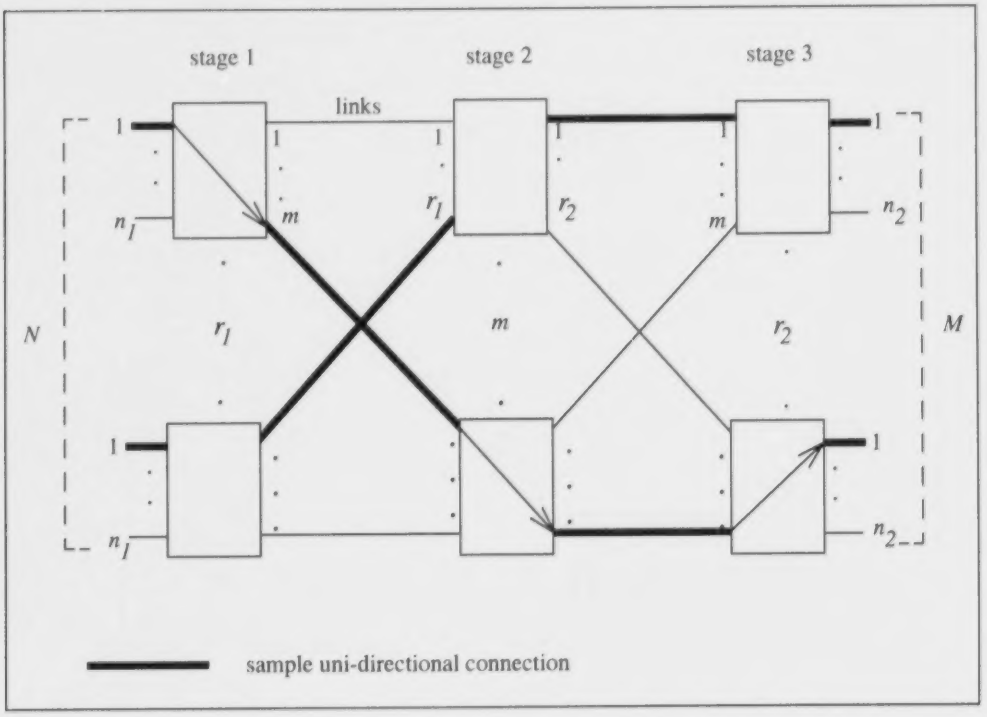


Figure 2.5. Three-stage double-sided switching network

2.4.5 ONE-SIDED MULTISTAGE NETWORKS

One-sided multistage networks are composed of one-sided switch modules arranged in multiple stages. Unlike double-sided networks, one-sided networks have one set of bi-directional network ports (N) that are used as inputs and outputs. The links connecting switch modules are also bi-directional.

A *two-stage single-sided network* [4], [24] is shown in Figure 2.6. The first stage switches appear to be double-sided with n input and m output ports. But in reality, the set

of ports on each first stage one-sided switch is split into two sets: network access and inter-switch trunking. There are n ports on each switch in the first stage used to provide user access to the switching network, and m ports used to interconnect the first and second stage switches, so the first stage switch must have a total of $(n+m)$ ports. It is important to understand the splitting of ports between access and trunking because this is basic to the optimization technique discussed later in the thesis.

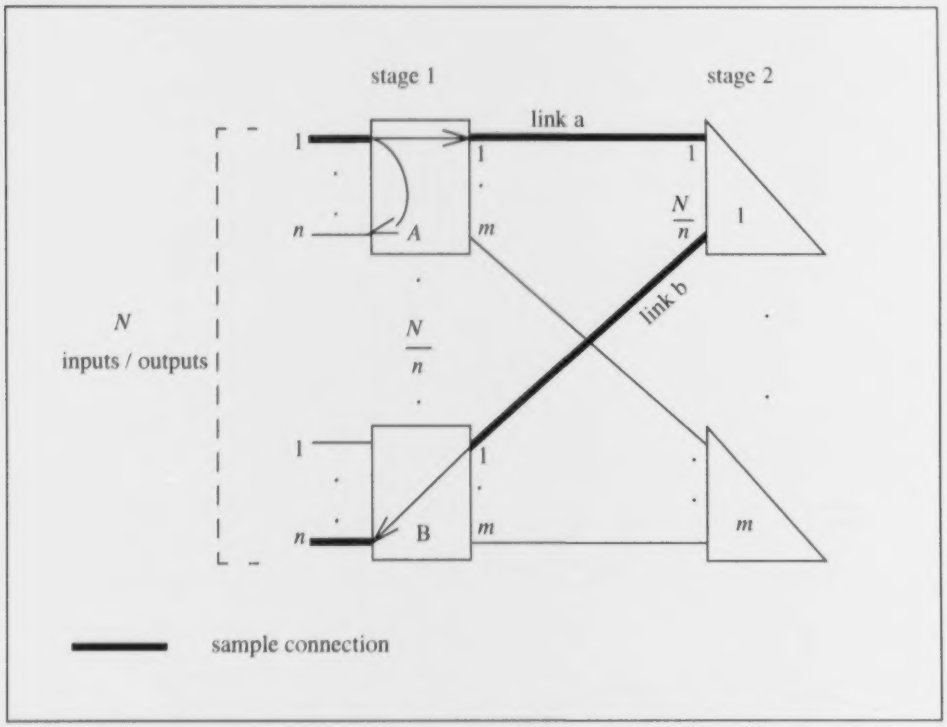


Figure 2.6. Two-stage single-sided switching network

A sample point-to-point connection is shown in Figure 2.6. The traffic originates from one of the idle ports on a first stage switch, passes through the second stage and then is routed back out to one of the idle ports on a first stage switch. A significant characteristic of such networks is the fact that traffic between ports on the same first stage switch does not travel through the second stage, but is routed internally through the first stage switch.

This is called a *deflected connection*. For example, a connection between ports *1* and *n* on switch *A* is made without using the second stage. Even though one-sided networks are inherently different from double-sided networks, there is one similarity: from the point of view of making a connection, the first stage in a two-stage one-sided network could also be perceived as the third stage as well as the first stage.

There are inherent differences between double-sided and single-sided networks on an implementation level. For example, since a double-sided network has a distinct set of inputs and output ports, there is no opportunity for deflection as is possible in a single-sided network.

2.4.6 BLOCKING IN MULTISTAGE NETWORKS

Even though a multistage switching network might be composed of non-blocking switching modules, the overall network is not guaranteed to be non-blocking. A multistage network is said to be *non-blocking* if any given set of idle ports can be interconnected, independent of the current state of the network. The request may be point-to-point or multicast in nature.

The ability of the switching network to set up a connection is dependant on the state of the network links. If one or more links required to set up a connection are already busy, then the request will be blocked. *Network blocking* is defined as the probability that a free input fails to connect to a free output due to the absence of an appropriate free link between stages. Looking at Figure 2.6, assume that all links between stages 1 and 2 are busy except for link *a* and link *b*. A connection between idle ports on switches A and B can be successfully completed over the path using link *a* and link *b*. If however, one or both of these links were busy, the connection would have been network blocked. Intuitively, a large number of second or middle stage switches in a multistage switching network implies lower network blocking because there will be more paths to choose from when making a connection. Thus there is an inherent trade-off between the number of switches in the middle stage and network blocking.

2.4.7 NON-BLOCKING SWITCHING NETWORKS

For certain applications requiring guaranteed service, networks have to be non-blocking for any connection request irrespective of existing connections. Clos [4] proved that the three-stage non-blocking network shown in Figure 2.5 is non-blocking for point-to-point traffic if:

$$m \geq n_1 + n_2 - 1 \qquad \text{(Eq. 11)}$$

The same condition is true for the two-stage one-sided network shown in Figure 2.6, with $n_1 = n_2 = n$. Clos' concept of non-blocking networks is very powerful. Since his original proposal, much work has been done in refining and extending his work to include multicast connections. Masson and Jordan [7] showed that the three-stage network shown in Figure 2.5 is non-blocking for broadcast traffic, assuming fan out switches at the first and third stages, if:

$$m \geq r_2 \cdot n_1 + n_2 - 1 \qquad \text{(Eq. 12)}$$

Equations 11 and 12 show that for a given multistage switching network, more middle stage switches give more possible routes from any input to one or more desired outputs, and reduce the probability of network blocking.

2.4.8 CHANNEL GRAPHS

The analysis of a multistage switching network often uses channel graphs. A *channel graph* is a graphical representation of the topology of a multistage network as seen by a call request originating at one of the first stage switches. Channel graphs have so far only been used for the analysis of point-to-point networks. The channel graph of a three stage switching network is shown in Figure 2.7.

Each point-to-point call originates through one of the ports on a first stage switch, has m possible link choices to go to the second stage, and then exactly one choice to connect to the desired output port on the output switch in the third stage from each second stage

switch. The same channel graph can be used to represent the two-stage single-sided network due to the fact that the first stage is used to both source and sink traffic.

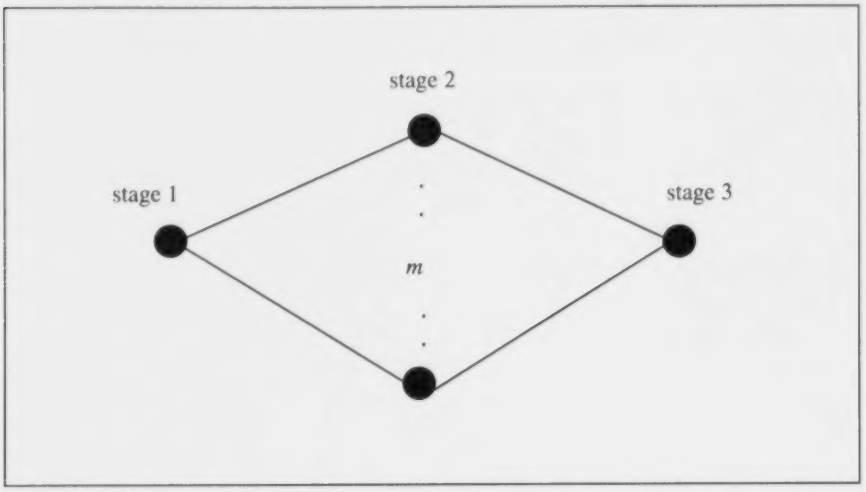


Figure 2.7. Channel graph of a three stage network

2.5 SUMMARY

In this chapter some basic terminology and concepts involved in the design of multistage fabrics have been outlined. The offered load and the desired grade of service are key specifications in determining the resources used in the design of a network. Resource sharing with full availability server groups allows for the provisioning of network resources, while incurring only a relatively small level of blocking. Erlang B and Engset traffic theories relate blocking, traffic offered, server group size and the number of sources.

3.0 NETWORK MODEL

This chapter presents the network model which serves as a framework for the design of the particular class of multicast networks on which we concentrate. The characteristics and the properties described in the model will be common to all of the test networks that we consider.

3.1 NETWORK TOPOLOGY

The type of networks we consider can be viewed in terms of an access component, and a network switching or inter-site trunking component, as shown in Figure 3.1. The building blocks of the network are DS-1 MCUs which provide the switching function.

3.1.1 ACCESS NETWORK

The *access network* consists of the user sites and the access links connecting them to the switching network. The user sites are assumed to be distributed over a wide area such that they can be grouped by location into a number of geographical zones. Since the intended service is videoconferencing, each site will have a room or studio equipped with *customer premises equipment* (CPE) consisting of audio and video equipment. Rather than limiting each site to one studio, we allow sites to have multiple studios where each studio is connected to the switching network via a DS-1 *access port*. For example, if the network is designed to provide multicast video-conferencing service for universities, then having only one studio for many faculties or departments would cause a lot of blocking. Thus, the number of access ports or studios required at a site is a design issue and is dependant on the amount of offered load at that site.

There are N sites in the network, each with p DS-1 access ports. An *access port* is bi-directional path allowing each studio to source and sink traffic from other sites in the network. Each port is connected to the switching network via a bi-directional *access link*. There is exactly one access link from each access port to the switching network. Thus, the switching network must provide access to a total of $(N \cdot p)$ links originating at user sites.

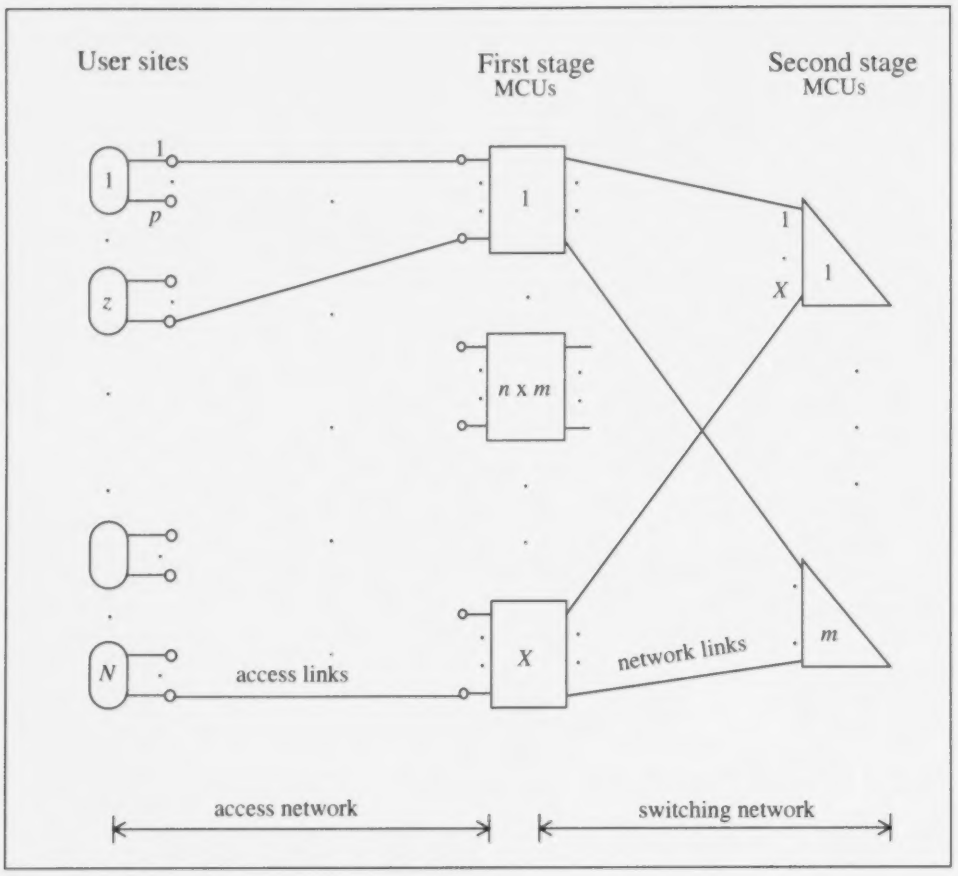


Figure 3.1. Network topology

3.1.2 INTER-MCU SWITCHING NETWORK

The switching network provides the necessary interconnectivity between studios at different sites in the network. The two-stage single-sided network described in Chapter 2 is proposed for this purpose. Each MCU has a maximum of C DS-1 ports. There are n ports on each first stage MCU connected to sites via access links, and m ports connected to second stage MCUs. That is, the first stage MCUs are partitioned to provide both access to user sites and inter-MCU trunking in a generalized $(n \times m)$ split, where $n + m \leq C$. All p access ports at a site connect to the same first stage MCU. For analytical purposes let us

assume there are z sites that have access links to the same first stage MCU, such that they completely fill the access ports of that MCU:

$$z = \frac{n}{p} \qquad \text{(Eq. 13)}$$

Having more than one site on a first stage MCU ($z > 1$) allows a direct connection between sites through their common first stage MCU. Such a connection does not involve the second stage and is said to be deflected at the first stage. Moreover, the first stage MCUs are assumed to have fanout capability. Any access port can connect to any number of free inter-MCU ports. This capability is used to set up multicast connections.

The number of first stage MCUs required, denoted by X, is related to the number of sites and the number of studios at each site. If each site in the network has p access ports, then a total of $N \cdot p$ access links need to be connected to the first stage of the network.

If n is the number of ports on each MCU for user access, then the number of first stage MCUs required is:

$$X = \left\lceil \frac{N \cdot p}{n} \right\rceil \qquad \text{(Eq. 14)}$$

The m network links at each first stage MCU form a distributed multistage switching fabric with one link between each pair of first and second stage MCUs. These bi-directional links are called network links. The number of MCUs in the second stage is equal to m with exactly m out of the available C ports at each MCU in use for inter-MCU trunking.

So far we have described the general topology of the multicast networks that will be considered. The network dimensions taken as parameters include the number of sites in a network, the number of access ports per site, the number of first and second stage MCUs required and the split between access and inter-MCU switching at the first stage MCUs. The MCUs are internally non-blocking and the first stage MCUs will perform the multicast fanout function. It is also assumed that connections requested between sites having access links to the same first stage MCU will be deflected and will not go through the second stage.

3.2 MULTICAST CALLING AND BLOCKING DEFINITIONS

Unlike point-to-point calls with one source-destination pair, multicast calls have one source and two or more destinations. The source and destinations for each call are assumed to be chosen uniformly from the N user sites in the network. Multicast call arrivals are assumed to be Poisson with negative exponentially distributed holding times. The number of destinations per call is a random variable, d, with distribution $P(d)$. Our analysis is for general $P(d)$, but in the absence of data on the number of destinations in a multicast call, we will assume a uniformly distributed number of destinations between $d = 1$ and D, inclusive. The average number of destinations per call is in general:

$$\bar{d} = \sum_{d=1}^{D} d \cdot P(d) \qquad \text{(Eq. 15)}$$

where $P(d)$ is the probability of d destinations per call and equals $1/D$ we have a uniform distribution. When $d = 1$ we have a point-to-point call.

To fully satisfy a multicast call request, the network should be able to connect the source to all d destinations. An individual *connection attempt* is defined to be an attempt to connect the source to one of its d destinations. In other words, a multicast call with d destinations implies d connection attempts between the source and the desired destinations as we have defined the word for present purposes. Because we consider networks with low but finite blocking, an important functional issue is how multicast calls are handled if one or more destinations are not reachable. We will assume that all feasible portions of multicast calls will proceed even if other portions are blocked. In concert with this, the event that is defined as multicast blocking is any unsuccessful connection attempt as defined above. That is, failure to connect any individual source-destination pair required as part of a multicast call will be recorded as an unsuccessful attempt. Hence we define multicast blocking to be the fraction of connection attempts that are unsuccessful.

3.3 MULTICAST CALL ROUTING MODEL

The choice of the routing model is an important consideration in the design of a network due to its impact on blocking performance. Our routing model is selected to minimize the use of network resources in setting up the requested connections. Successfully connecting a multicast call requires more network resources than connecting a point-to-point call and therefore the choice of a good multicast routing model is more significant in a network designed to carry multicast traffic.

Each multicast call consists of a source, s, and d destination sites. The first stage MCU providing access to the source site will be called the source MCU. MCUs connected to the access ports of the destination sites are called destination MCUs. The source MCU is the signal fanout point at which all required multicast copies are spawned and to which all the destinations are connected in a star.

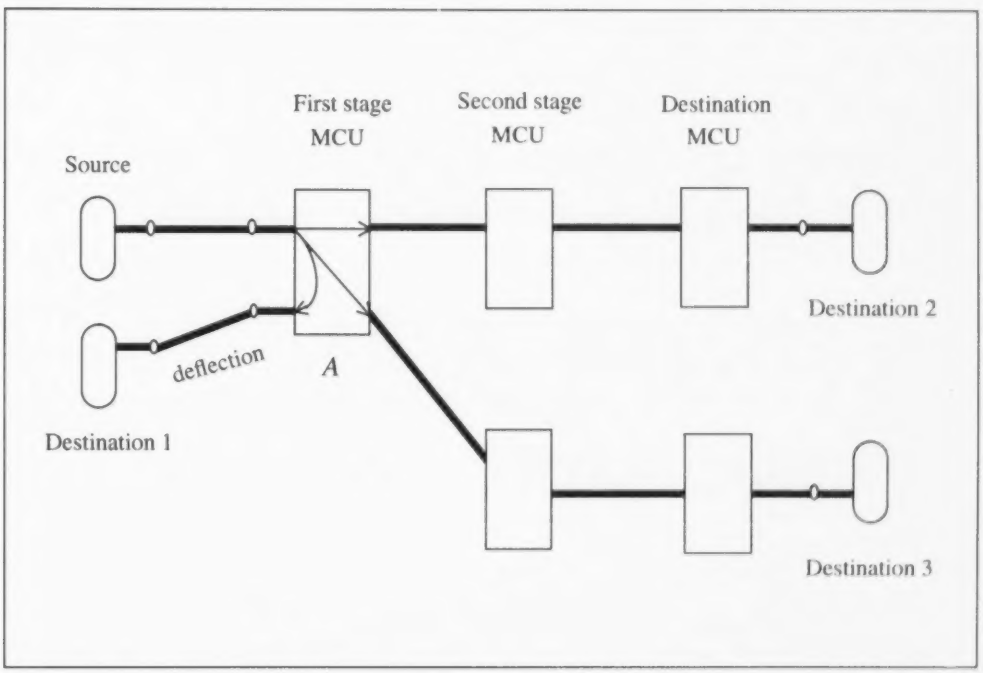


Figure 3.2. Multicast routing model including deflection

Figure 3.2 illustrates the ways a first stage MCU may be connected to the source and destinations in a multicast call. The first stage MCU generates the fanout as required for the complete multicast call. In Figure 3.2, destination 1 has access ports connecting to the same first stage MCU as the source and is therefore connected by *deflection* at *A*. Destinations 2 and 3 are on first stage MCUs different than the source MCU and are therefore routed through the second stage. Each connection between the source and destinations 2 and 3 passes through 3 MCUs: the source MCU, a second stage MCU and then the destination MCU. Note that the destination MCUs are in fact first stage MCUs. Thus the multicast signal enters the switching network through the source MCU in the first stage and exits the switching network via the destination MCUs which are also in the first stage. In case of deflection, the source and the destination MCU are the same.

3.4 SUMMARY

The type of networks we consider can be viewed in terms of an access component and a network switching component. The access component consists of user sites and access ports. Multiple access ports at a site allow for multiple simultaneous connections receiving or originating traffic at a site. Access ports are connected to the switching network via bi-directional links called access links. The switching network is a two-stage-single sided network with MCUs as the switching modules. The ports on each first stage MCU are partitioned between user access to the switching network, and inter-MCU junctoring. Ports on second stage MCUs are used solely for junctoring. Multicast calls have Poisson arrivals, and each call has a source site and one or more destination sites. The number of such destinations is uniformly distributed on [1.,.D]. The network attempts to connect the source to all of the destinations requested in the call. A failure to connect the source to one of the destinations is considered a multicast blocking event. It is assumed that all feasible portions of a multicast call will proceed even if some destinations are blocked. In order to support multicasting, the first stage MCUs are assumed to have fanout capability. The first stage MCU to which the source is connected is the signal bridging point to which all destinations are connected in a star. A connection between a source-destination pair is deflected if the source and the destination are on the same first stage MCU, in which case the connection is set up without using the second stage. If the destination has its access ports on

an MCU different from the source MCU, the connection is set up through the second stage utilizing two network links.

4.0 ANALYTICAL MODEL

This chapter presents an analytical model for the blocking performance of a multicast network as previously described. The analysis predicts the end-to-end blocking in a network as a function of multicast traffic levels and network dimensions. The analysis that now follows is one of the main contributions of this work. It involves the combining and extension of channel graph and conventional full-availability traffic theory to obtain an original and useful analytical model for the previously described class of networks.

4.1 BLOCKING CHARACTERISTICS

To calculate the end-to-end blocking in a multicast network, we must understand the blocking characteristics of the network model described in the previous chapter. We defined multicast blocking as the failure to connect any one of the source-destination pairs which are a part of a multicast call. There are two blocking mechanisms that contribute to the overall multicast blocking: we call them *site* or *access blocking* and *inter-MCU link* or *network blocking*. Let the site congestion, P_s, be the probability of time congestion in access ports at a site, preventing a site from establishing a requested connection. Let the network blocking, P_n, be the probability that an individual connection attempt from a source MCU to a destination MCU is blocked in the two-stage inter-MCU junctoring network.

The *end-to-end blocking*, P_{end}, is then defined for the path from an originating source to each one of its destinations, considered individually. From the point of view of trying to make a connection between a source site and a destination site, there are three points in the network along that path that can lead tc blocking (see Figure 4.1). First, all p access ports at the source could be in use. Second, the switching network may be unable to establish a connection between the source and the destination MCU due to the unavailability of one or more network links required for the connection. Lastly, all p access ports at the destination site could be in use.

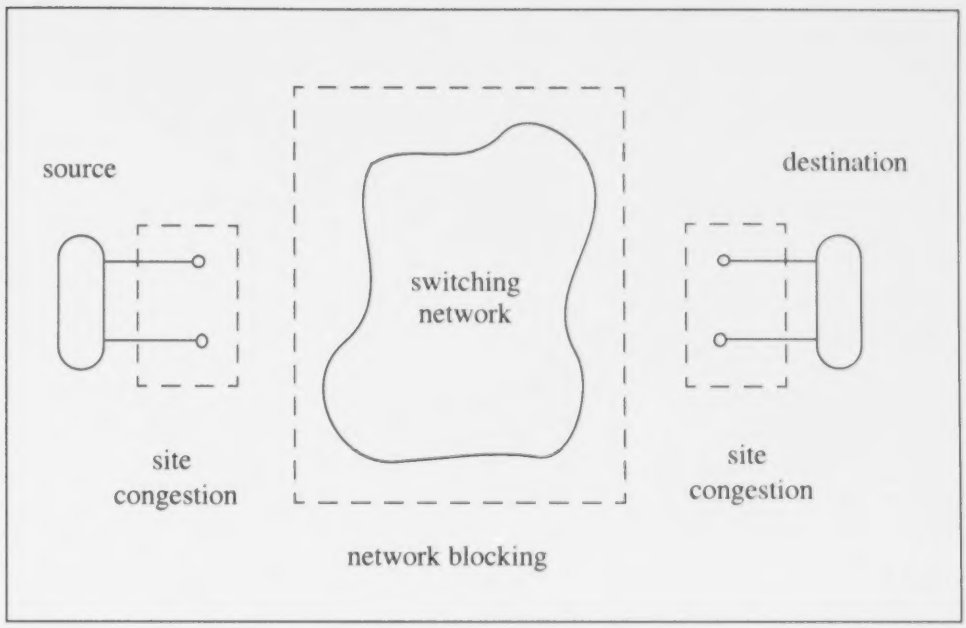


Figure 4.1. End-to-end blocking model

Therefore, in general:

$$P_{end} = 2 \cdot P_s + P_n \qquad \text{(Eq. 16)}$$

For the special case of a deflected connection, $P_n=0$ as no attempt is made on the switching network. Also note that if the source of a call is site-blocked, no connection attempts are made and hence no additional offered traffic is presented to the switching network.

4.2 STRUCTURE OF THE ANALYTICAL MODEL

Before we derive the general equations that form a calculable model for P_{end}, we will present an overview of the basic interrelationships involved in the analysis that follows and define some of the key parameters (see Figure 4.2). The procedure will be to calculate the total traffic that is offered to each site in the network and then carry the analysis through the access ports and on to the switching network. We will let A_o be the offered traffic *originating* from each site. To this will be added the offered terminating traffic to

calculate the total site blocking, P_s, at the site using Erlang theory. The actual traffic carried by each access port is the *access link* or *port utilization*, a_c. There are n access links with utilization a_c that offer traffic to each first stage MCU. But since some of the traffic is deflected, we must take into account the probability of a connection attempt being deflected, P_{def}. The traffic that is actually offered to the switching network is then used to calculate the network blocking, P_n, using channel graph analysis extended for multicast considerations.

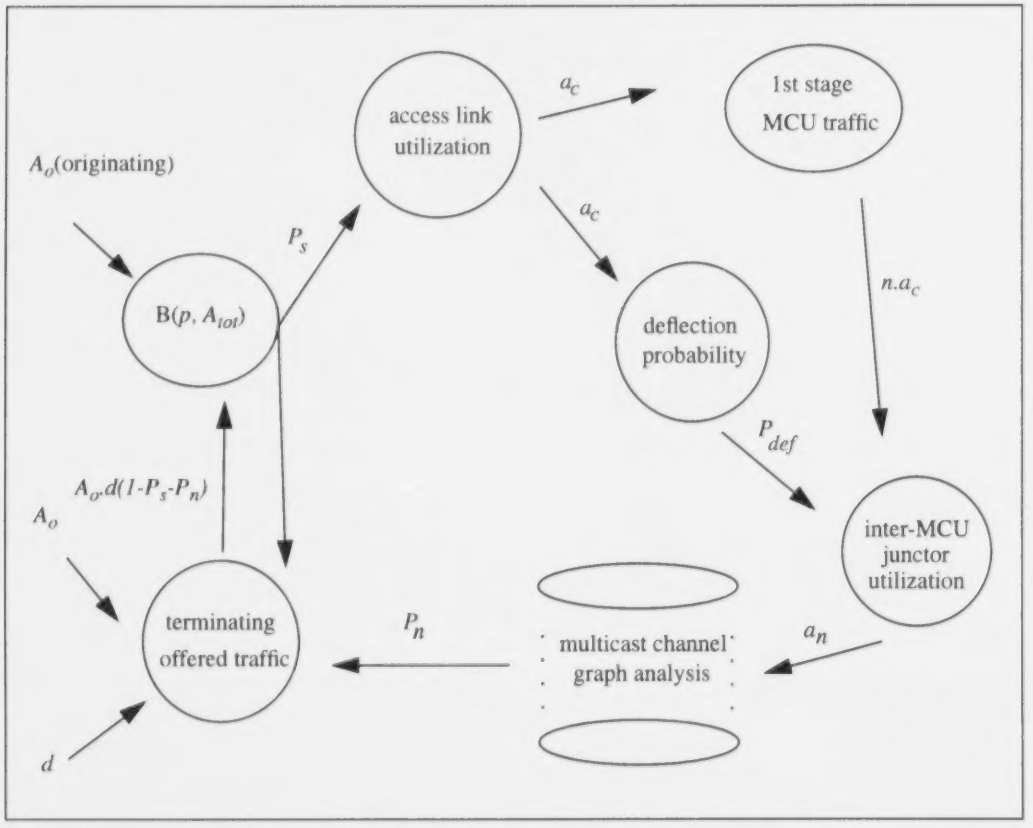


Figure 4.2. Functional Interrelationships of the Analytical Model

4.3 OFFERED TRAFFIC

In the network model proposed, any site can originate and terminate traffic via its access ports. Multicast traffic is comprised of calls consisting of a source site and d randomly chosen destination sites with each site appearing only once in a call. If users at each site have the capacity to originate multicast calls with intensity A_o to d uniform randomly distributed destinations per call, then the total offered traffic at a site is the sum of the traffic sourced and that fraction of all other offered traffic for which: (a) the site is a destination, (b) the terminating connection attempt is not blocked at the originating access group and (c) the terminating connection attempt is not blocked in transiting through the switching network. Because the originating group, network and terminating group in any path are statistically independent, time congestion in the access groups will be lowered to the extent that offered terminating traffic intensity is reduced from network or originating site blocking. Therefore, the total offered traffic intensity on each site is itself dependent on the blocking of the network and access groups themselves.

In a multicast environment, each site is more likely to be involved in terminating a call than originating one, if all sites are equally likely as destinations. Therefore, if, initially we neglect blocking considerations, each of the N sites in the network originates A_o erlangs to d destinations, we can expect ($N \cdot A_o \cdot d$) total erlangs of terminating offered traffic to be generated by all sites, of which each site will receive ($1/N$), if all destinations are equally likely. The result is that the total offered terminating traffic at a site, neglecting blocking, is ($A_o \cdot d$). Bringing all the factors together, including blocking, the total offered traffic for each site in the multicast environment with d destinations per call is the sum of traffic sourced by a site and the traffic terminated at the site:

$$A_{tot}(d) = A_o + A_o \cdot d \cdot (1 - P_s - P_n) \qquad \text{(Eq. 17)}$$

where P_s is the blocking of the access group and P_n is the network blocking of each connection attempt in a multicast call. Since d is more generally a random variable with distribution $P(d)$, the average traffic offered by a site is:

$$A_{tot} = \sum_{d=1}^{D} A(d) \cdot P(d) \qquad \text{(Eq. 18)}$$

We will assume $P(d) = 1/D$ in our subsequent calculations.

From now on we will consider the offered traffic as a function of the number of destinations per call, d. The probability of site congestion and network blocking will be derived as functions of d and the average end-to-end blocking with a general distribution for d will be:

$$P_{end} = \sum_{d=1}^{D} (2 \cdot P_s(d) + P_n(d)) \cdot P(d) \qquad \text{(Eq. 19)}$$

The explicit functional dependence of the analysis below is dropped in the analysis that follows, for simplicity.

4.4 SITE CONGESTION, P_s

Site congestion is defined as the probability that all access ports at a site are in use. The access ports are analogous to a full availability server group that is shared by the users at a site. We assume that the number of users at a site is large compared to the number of access ports, and that fully blocked multicast calls are strictly cleared. Thus, the probability of site congestion can be modelled using the Erlang B formula. A total of A_{tot} erlangs is offered to p access ports at a site. Therefore, the site congestion probability is:

$$P_s = B(p, A_{tot}) = \frac{(A_{tot})^p / p!}{\sum_{i=0}^{p} (A_{tot})^i / i!} \qquad \text{(Eq. 20)}$$

For the Erlang B model, time congestion equals call congestion. Time congestion is the average fraction of time for which all ports at a site are in use whereas call congestion is the fraction of the total number of connection attempts that are blocked due to unavailability of a free access port at a site. Thus, one test of whether Erlang B applies is to see

whether average time congestion equals the average call congestion. This test will later be done with the emulator discussed in Chapter 5.

4.5 ACCESS LINK/PORT UTILIZATION, a_c

The total offered traffic, A_{tot}, as calculated in Equation 17, is subject to site blocking. The total traffic carried out of a site is borne by the p access ports (see Figure 4.3). Therefore, the traffic *carried* by each access port and each access link, is:

$$a_c = \frac{A_{tot} \cdot (1 - P_s)}{p} \qquad \text{(Eq. 21)}$$

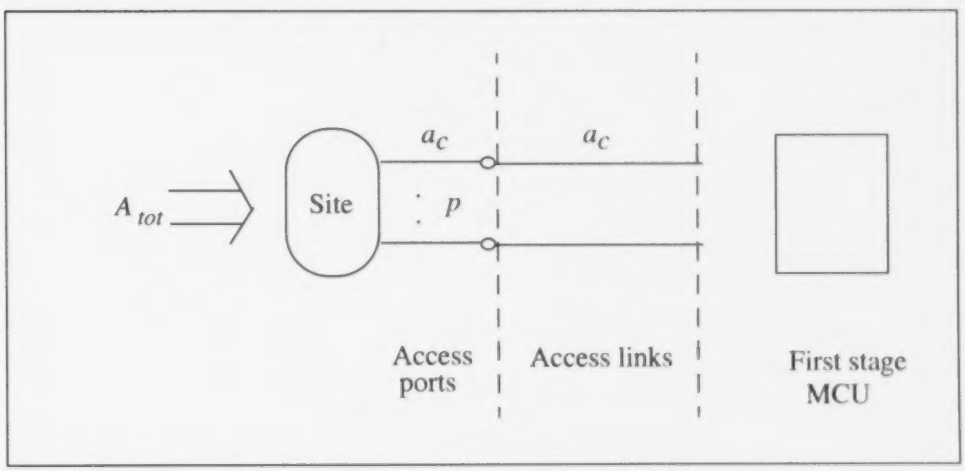


Figure 4.3. Access port utilization

Given a_c, we will need to know the probability that exactly k of the n access links at a first stage MCU are busy. This can be based on the binomial distribution:

$$P(k) = \binom{n}{k} \cdot (a_c)^k \cdot (1 - a_c)^{n-k} \qquad \text{(Eq. 22)}$$

The k busy links can be chosen in $\binom{n}{k}$ ways and the probability of all k being busy is $(a_c)^k$; the remaining $(n\text{-}k)$ links are free with probability $(1\text{-}a_c)^{n-k}$.

4.6 DEFLECTION PROBABILITY, P_{def}

A total of d connection attempts are made per call. We define deflection probability as the fraction of successful connection attempts that are deflected. The probability of a connection being deflected (see Figure 4.4) can therefore be calculated by taking the ratio of the expected number of deflection destinations per call to the total number of connected attempts per call:

$$P_{def} = \frac{E_{def}}{d(1 - P_s - P_n)} \qquad \text{(Eq. 23)}$$

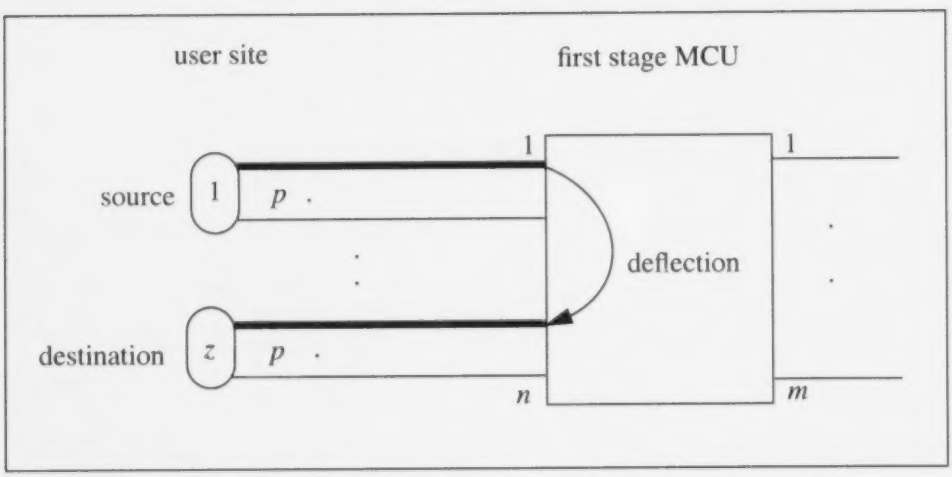


Figure 4.4. Deflected connection

According to the network model, the number of sites having access to the same first stage MCU is z. Since the source is fixed, a maximum of $(z\text{-}1)$ deflections can occur per multicast call. Thus, the expected number of deflected connections in a multicast call is:

$$E_{def} = \sum_{i=1}^{z-1} i \cdot P_{def}(i|d) \qquad \text{(Eq. 24)}$$

where $P_{def}(i|d)$ is the probability of exactly i connection attempts being deflected given a call with d destinations. The probability of a deflected connection is not a constant, but depends on the current number of deflections completed in the call. This is illustrated in Example 4.1, with reference to Figure 4.4.

Example 4.1 Suppose that the network has $N=18$ sites, and the number of sites connected to the same first stage MCU is $z=3$. Consider calls with $d=2$ destinations. What is the probability that both destinations are deflected?

Solution: Once the source is fixed, there are 2 sites left with access ports on the same source MCU. The probability of one deflected attempt is 2/17 and the probability of a second deflection is 1/16. Therefore, the probability of two deflections is $(2/17 \cdot 1/16)$.

More generally, $P_{def}(i|d)$ is based on sampling without replacement using the hypergeometric distribution to arrive at the following equation:

$$P_{def}(i|d) = \frac{\binom{z-1}{i} \cdot \binom{N-z}{d-i}}{\binom{N-1}{d}} \cdot (1-P_s)^i \qquad \text{(Eq. 25)}$$

There are $\binom{z-1}{i}$ ways of picking i deflected destinations from the $(z\text{-}1)$ which have access links to the source MCU, and $\binom{N-z}{d-i}$ different ways of picking $(d\text{-}i)$ non-deflected destinations in the call out of the $(N\text{-}z)$ destinations that do not have access links to the source MCU. The product of these two terms is the number of ways i deflected destinations and $(d\text{-}i)$ non-deflected destinations can be chosen. Dividing by $\binom{N-1}{d}$, which is the number of ways d destinations can be chosen after the source has been designated, gives the probability that i connections can be deflected if not site blocked. A deflection opportunity is actually completed if the destination has at least one access port free.

Therefore, multiplying by $(1-P_s)^i$ gives the probability of exactly i deflection opportunities being established.

4.7 NETWORK LINK UTILIZATION, a_n

So far we have considered the flow of traffic from the user sites onto the access ports and to the first stage MCU. Some of the traffic offered to the first stage via the access links is deflected. The rest is offered to the switching network to be carried through the second stage using the network links and back to the destination MCUs. It is this network offered traffic that causes network blocking. Therefore we are interested next in calculating the utilization of the network links.

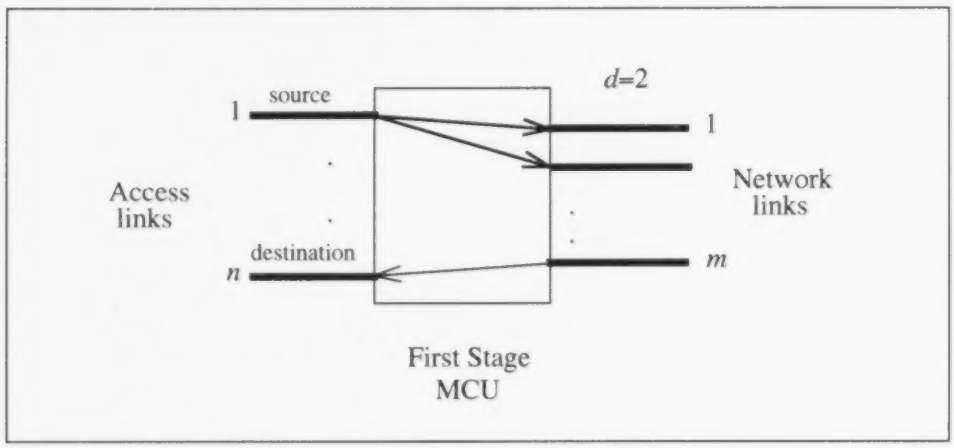


Figure 4.5. Multicast traffic amplification at first stage MCU

Consider the ports at a first stage MCU. Each first stage MCU has a limited total number of ports which are split to provide access and inter-MCU junctoring. The access side of the MCU gathers n access links each with utilization a_c as calculated in section 4.4. The calculation of a_n is more involved due to the multicasting that occurs at a first stage MCU and the multistage switch nature of the blocking that arises in the network, as opposed to the full availability group blocking in the access. In particular, a traffic carrying access link has a different effect on the network link load depending on if it is busy as

a source or as a destination. If a busy access link is associated with a source site, then its traffic effect is magnified at the first stage MCU to a total of d network links out of m, whereas if it is busy as a destination it only connects to one network link (see Figure 4.5). Therefore, the access link utilization must be the summation of the fraction of the time an access link is busy connecting to a source site, a_s, and the fraction of the time it is busy connecting to a destination site, a_d:

$$a_c = a_s + a_d \qquad \text{(Eq. 26)}$$

However, Equation 26 can be re-written as:

$$a_c = (a_c \cdot p_s) + (a_c \cdot p_d) \qquad \text{(Eq. 27)}$$

where p_s and p_d are the probabilities of a site being in use as a source or destination respectively, given that the site is in use:

$$p_s + p_d = 1 \qquad \text{(Eq. 28)}$$

Since each multicast call has d destinations, an access link is d times more likely to be busy as a destination, so:

$$p_d = d \cdot p_s \qquad \text{(Eq. 29)}$$

A total of n access links offer traffic to each first stage MCU which is then spread over m network links (see Figure 4.5). Therefore, we multiply the access link utilization by n and divide by m to calculate the network link utilization. The first term is also multiplied by d due to the effects of multicasting:

$$a_n = [a_c \cdot p_s \cdot d + a_c \cdot p_d] \cdot \frac{n}{m} = a_c [p_s \cdot d + p_d] \cdot \frac{n}{m} \qquad \text{(Eq. 30)}$$

So far however, Equation 30 is based on a value of a_c that includes the probability of an access link being in use in a deflected connection. But a deflected connection contributes nothing to the network link utilization so it must be excluded from the calculation of a_n as follows:

$$a_n = a_c \cdot (1 - P_{def}) \cdot [p_s \cdot d + p_d] \cdot \frac{n}{m} \qquad \text{(Eq. 31)}$$

Equation 31 now gives the average utilization of a network link connecting a first and a second stage MCU.

4.8 NETWORK BLOCKING, P_n

P_n is the fraction of individual connection attempts which encounter blocking in the two-stage inter-MCU network. There is a difference between the analysis of network blocking in terms of point-to-point traffic and multicast traffic. Multicast traffic uses more network resources on a per call basis due to the multicast bridging that occurs at the first stage. On a per call basis assuming zero end-to-end blocking and zero deflection, point-to-point traffic uses 2 network links between the first and the second stage, whereas a multicast call with d destinations uses $(2 \cdot d)$ network links. Therefore, we consider the case of point-to-point traffic before extending it to multicast traffic.

4.8.1 NETWORK BLOCKING FOR POINT-TO-POINT TRAFFIC

In a point-to-point traffic environment, there is one source and one destination per call. If a call is not site blocked or deflected, the source MCU will request a connection to the destination MCU through the remaining two-stage switching network. Channel graphs are used to analyse the blocking in the switching network. A channel graph is a simple graphical representation of a switching network. It is a generic illustration of all possible routing options available to a call request and is dependant on the linking pattern between the different stages of the multistage switching network. The channel graph of the two-stage single sided switching network used in this work is shown in Figure 4.6 for point-to-point traffic.

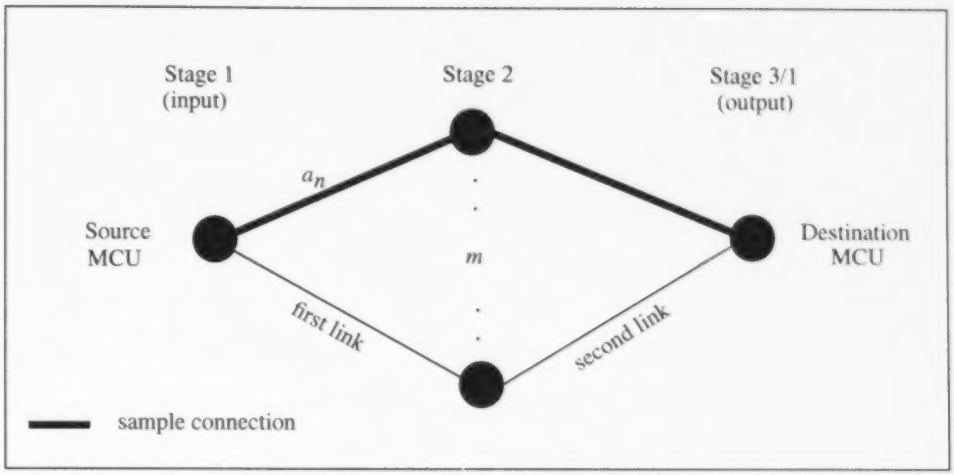


Figure 4.6. Channel graph for point-to-point traffic

Figure 4.6 shows three different stages because from the point of view of making a connection, we have to traverse three different MCUs: the source MCU, a second stage MCU and the destination MCU. If not deflected or site blocked, each point-to-point call requests a connection between the source MCU and the destination MCU. These are both in the first stage. There are m possible ways to get to a second stage MCU from the source MCU. There is one way to get to the destination MCU from each second stage MCU. Thus, 2 network links are used to complete a connection.

There are a total of m different possible paths in parallel from the source MCU to the destination MCU. Each path consists of 2 links, one connecting the source MCU to the second stage and the other connecting the second stage MCU to the destination MCU. A connection attempt will be blocked if one or both links in all m paths are busy. Looking into a path from the source MCU, if the first link is busy, then the path is blocked no matter what the state of the second link is. On the other hand, if the first link is free, then the path is blocked if and only if the second link is busy. Note that each link in the switching network is busy with probability a_n and free with probability $(1\text{-}a_n)$.

Therefore, if exactly k of the m first stage links in the channel graph are free, then the probability of second stage blocking is $a_n^{\,k}$ as all k second links connecting to those

free links must be busy for the call to be blocked. The binomial distribution can be used to calculate the probability that k of the m first stage links are free, because these links are used independently. Thus the probability of network blocking for point-to-point traffic is:

$$P_n(d=1) = \sum_{k=0}^{m} \left[\binom{m}{k} \cdot (1-a_n)^k \cdot a_n^{m-k} \right] \cdot a_n^k \qquad \text{(Eq. 32)}$$

where $\left[\binom{m}{k} \cdot (1-a_n)^k \cdot a_n^{m-k} \right]$ is the probability that k network links at the source MCU are free and $(m-k)$ are busy, and a_n^k is the probability that all k second links connecting to the free links at the source MCU are busy.

The channel graph shown in Figure 4.6 can be used in another more familiar method to calculate the probability of a connection attempt being network blocked for point-to-point traffic. If $(1-a_n)$ is the probability of a link in a path being free, then the 2-link path is free with probability $(1-a_n)^2$ and blocked with probability $[1-(1-a_n)^2]$. Since there m such parallel paths to get from the source MCU to the destination MCU, the probability of network blocking can also be written as:

$$P_n(d=1) = \left[1-(1-a_n)^2\right]^m \qquad \text{(Eq. 33)}$$

Equations 32 and 33 are equivalent in terms of calculating the probability of network blocking in a point-to-point traffic environment. The point-to-point analysis shows that network blocking is inversely related to the number of middle stage MCUs, m. Having more second stage MCUs provides more routing options for a connection attempt thereby reducing the probability of network blocking. This is true in a multicast environment as well, but the analysis to calculate multicast network blocking is more complex as discussed in the next section.

4.8.2 MULTICAST NETWORK BLOCKING

The multicast fanout that occurs at the first stage source MCU has a significant impact on network performance because more network links must be allocated to each call. The

source MCU requests a connection to d destination MCUs through the switching network and we want to calculate the fraction of the connection attempts that are blocked by the switching network. The probability of a connection attempt being blocked can be calculated by taking the ratio of the expected number of blocked attempts to the actual number of attempts made on the switching network for each call:

$$P_n = \frac{\sum_{j=1}^{d} j \cdot P_n(j|d)}{d \cdot (1 - P_s - P_{def})} \qquad \text{(Eq. 34)}$$

where $P_n(j|d)$ is the probability that exactly j out of d connection attempts in a multicast call are network blocked. The denominator is the total number of attempts made on the switching network for one multicast call.

To obtain $P_n(j|d)$ we consider the channel graph for the two-stage switching network extended for multicast considerations. We are unaware of any work relating to the use of channel graphs to calculate blocking in a multicast environment. The multicast channel graph of Figure 4.7 is similar to the channel graph of Figure 4.6 except that the source MCU is requesting a connection to d destination MCUs instead of one.

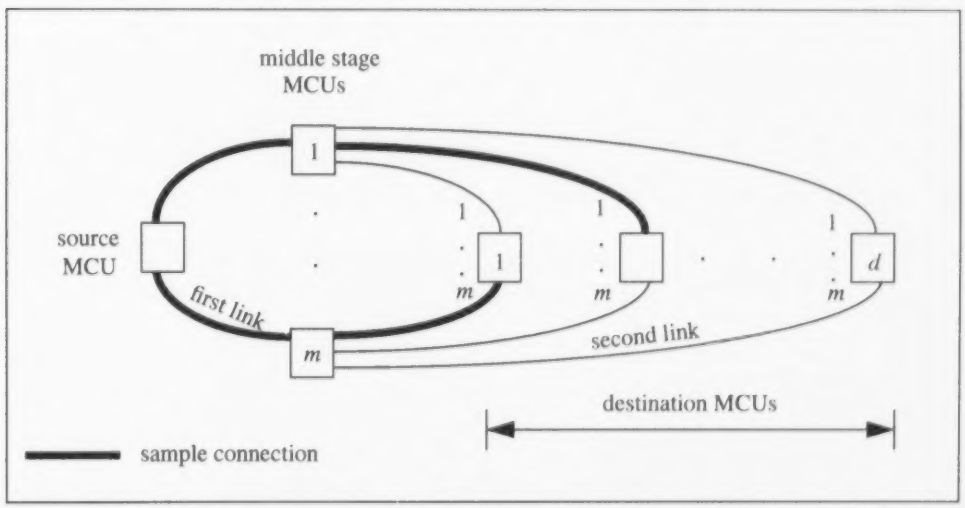


Figure 4.7. Multicast channel graph for a multicast call request

In Figure 4.7, the source MCU has m links to get to the second stage as in the point-to-point case, but now has to connect to d destinations via different first stage links. Each destination MCU has one link connecting it to each of the m middle stage MCUs. Obviously, if $d>m$, blocking occurs. Note that even if multiple destinations terminate on the same third stage MCU, the way this network operates requires a separate connection for each one from the first stage MCU

Conceptually, the multicast network blocking analysis is based on the point-to-point case. We are interested in all cases in which at least one of the destinations is network blocked. For each case, we look at the states of the network links that result in exactly j destinations being network blocked. As before, a destination will be blocked if one or both of the links in a potential path are busy and therefore unavailable to complete the connection. The number of destinations that can be blocked, j, ranges from *1* to d. If j destinations out of d are blocked, then $(d-j)$ destinations are successfully connected. This implies that at least $(d-j)$ first links connected to the source MCU have to be free. Thus, the probability of that j connection attempts out of a possible d are network blocked in a multicast environment can be written as:

$$P_n(j|d) = \binom{d}{j} \cdot \sum_{k=d-j}^{m} P_f(k) \cdot \beta(k)^j \cdot (1-\beta(k))^{d-j} \qquad \text{(Eq. 35)}$$

where $\binom{d}{j}$ is the number of ways of choosing j blocked destinations out of d, $P_f(k)$ is the probability that k first links out of m are free given d destinations per call, and β is the probability of one connection attempt being blocked. There are two terms in the above equations, $P_f(k)$ and $\beta(k)$ that are unknowns and they are derived in the following.

4.8.2.1 PROBABILITY OF NETWORK LINKS BEING FREE, $P_f(d,k)$

Unlike the access links or the network links in the case of point-to-point analysis, the probability that a given number of network links are free in the multicast case, $P_f(k)$, is not simply binomial. This is because of the signal bridging that occurs in the first stage. As a

result the system states of the network links at a first stage MCU are rather correlated. For example, if $d=3$ there will be a heightened non-binomial probability of seeing 0, 3, 6, 9 links in use together, rather than a binomial progression if link states were independent. Conversely 2, 4, 5, 7 links in use out of a 1st stage MCU cannot happen if $d=3$ exactly and there is no site blocking. The Markov chain shown in Figure 4.8 highlights the transitions between the system states of the network links for $d=3$ and $m=6$ and illustrates the restrictive nature of the Markov chain due to multicasting. States 1, 2, 4 and 5 are not allowed unless one or more connection attempts are blocked or deflected. The transition from 0 to 3 is more likely than transitions from 0 to 2 with one blocked attempt or from 0 to 1 with 2 blocked attempts.

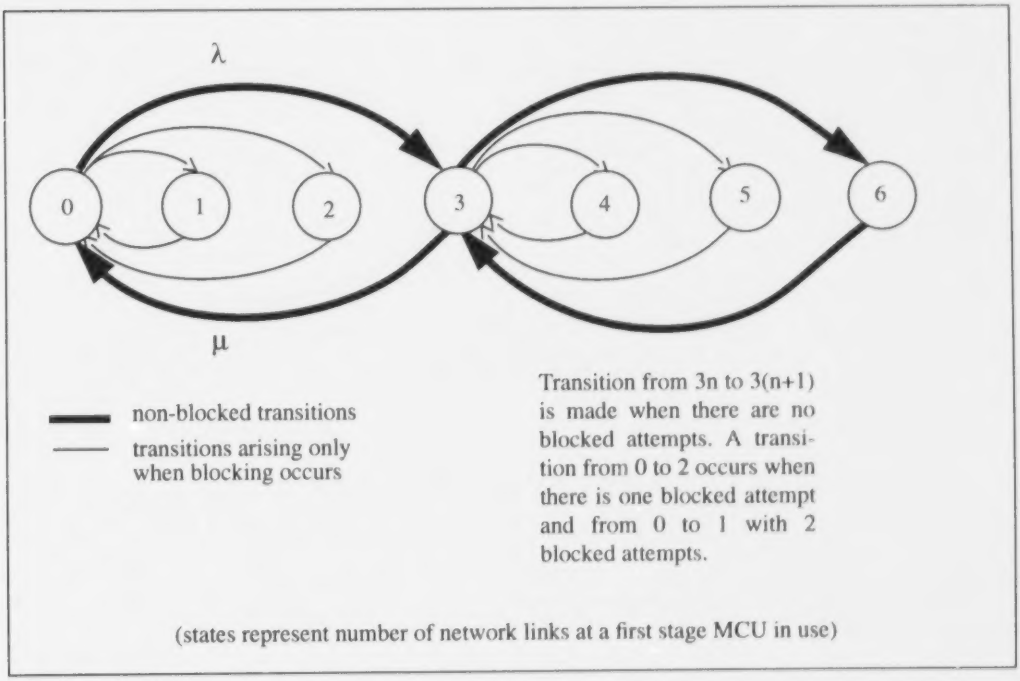

λ
0
1
2
3
4
5
6
μ
non-blocked transitions
transitions arising only when blocking occurs
Transition from 3n to 3(n+1) is made when there are no blocked attempts. A transition from 0 to 2 occurs when there is one blocked attempt and from 0 to 1 with 2 blocked attempts.
(states represent number of network links at a first stage MCU in use)

Figure 4.8. Markov Chain for network link states ($d=3$, $m=6$)

The hypothesis that the busy states of the network links are not binomially distributed is later validated by simulation results. The problem of deriving the distribution of the busy states of the network links still remains. We want the probability that k out of m network links are busy, while recognizing the d-way fanout effect just discussed. One way

to approach this is to recognize the dependence between the usage of access links on the user side of a first stage MCU and network links on the inter-MCU network side. In the simplest case, if only one network link is in use at a first stage MCU, then one access link has to be busy on that MCU. On the other hand, if $d=2$ and 2 network links are busy at a first stage MCU, then the number of access links that result in 2 network links being busy is dependant on how the access links are being used. One access link busy as a source could result in those 2 network links being busy due to multicast bridging. Alternately, two access links busy as destinations would also result in 2 network links being busy on the same MCU. Since the probability that k out of n access links are busy can be predicted using the binomial formula, we can derive the distribution of the busy states of the network links by mapping the access links states onto those of the network links. This will allow us to derive equations to calculate the probability that k of the m network links are busy at a first stage MCU, $P_b(k)$. The probability that k network links are free is equal to the probability that $(m-k)$ network links are busy:

$$P_f(k) = P_b(m-k) \qquad \text{(Eq. 36)}$$

The basic concept is to determine the states of the access links that result in k network links being busy. The link states on the network side are the independent variables and we look to corresponding access side states to calculate the probability of k network links being busy. The analysis is not trivial because of the dependence of the state of the access links on the way they are actually used. Therefore, we have to look at different cases depending on access link usage and then look within each case at all possible states of the access links that result in k network links being busy. The probability distribution of the busy states of access links as given by Equation 22 cannot be used because the access link utilization a_c also has contributions from deflected connections. Deflected connections do not effect the network links and therefore must not be included. The actual access link utilization that will be used is discounted by the probability that the link is in use by a deflected connection, so that the access link utilization without deflection is:

$$a'_c = a_c \cdot (1 - P_{def}) \qquad \text{(Eq. 37)}$$

Therefore, the probability that i out of n access links are busy such that they have an effect on the network links is given as:

$$P_{ac}(i) = \sum_{i=0}^{n} \binom{n}{i} \cdot \left(a'_c\right)^i \cdot \left(1 - a'_c\right)^{n-i} \qquad \text{(Eq. 38)}$$

The probability that an access link is used to connect to a source or a destination site is defined as p_s and p_d respectively. To simplify what follows, we refer to an access link being busy as a source or a destination, rather than saying that an access link is busy because it is connected to a source or a destination. Let Φ be the number of ways in which the access links of an MCU can be configured to result in k links busy on the network side of the first stage MCU. In the above example with d=2, there are 2 ways that 2 network links can be made busy: one access link busy as a source with probability p_s, or 2 access links busy as destinations each with probability p_d. In general let κ be the number of access links busy as sources and τ be the number of access links busy as destinations. The total number of access links busy for a particular configuration that result in k network links being busy is $\kappa + \tau$. We go through all possible busy states of the network links from 0 to m and derive equations for all Φ cases. There are 4 scenarios depending on the number of busy network links, k, given d destinations per multicast call. For clarity, we will first derive the mapping equations between input and output MCU link states assuming no blocking. The blocking correction for each scenario then follows.

1. Zero network links busy (k=0)

If zero network links are busy at a first stage MCU, that means that no access links are busy on the user side of that MCU. Therefore, the probability of zero network links busy is equal to the probability of zero access links being busy as defined above in Equation 38:

$$P_b(0) = P_{ac}(0) \qquad \text{(Eq. 39)}$$

2. $1 \le k \le d-1$

If the number of busy network links is less than d, then the access links must all be busy as destinations and none are busy as sources. This is because an access link busy as a source would result in d network links being busy due to multicasting. In this case the number of busy network links is equal to the number of busy access links. Therefore, the probability that k network links are busy is:

$$P_b(k) = P_{ac}(k) \cdot (p_d)^k \qquad \text{(Eq. 40)}$$

where the right hand side is the probability of k access links which are independently busy as destinations.

3. $k = d$

Neglecting the possibility of blocking, there are 2 ways in which the access links can be configured such that the number of network links busy is equal to the number of destinations per call. In the first, one access link is busy as a source and this results in d network links being busy. In this configuration $\kappa_1 = 1$ and $\tau_1 = 0$. The second configuration has d access links being busy as destinations, also resulting in d network links being busy. Here $\kappa_2 = 0$ and $\tau_2 = d$. The probability that exactly d network links are busy is:

$$P_b(d) = \sum_{i=1}^{2} P_{ac}(\kappa_i + \tau_i) \cdot p_s^{\kappa_i} \cdot p_d^{\tau_i} \qquad \text{(Eq. 41)}$$

where $\kappa_i + \tau_i$ is the number of access links busy in configuration i.

4. $d+1 \le k \le m$

For this case the probability that k network links are busy can be written as:

$$P_b(k) = \sum_{i=1}^{\Phi} \binom{\kappa_i + \tau_i}{\kappa_i} \cdot P_{ac}(\kappa_i + \tau_i) \cdot p_s^{\kappa_i} \cdot p_d^{\tau_i} \qquad \text{(Eq. 42)}$$

where $\binom{\kappa_i + \tau_i}{\kappa_i}$ is the number of ways of choosing κ_i access links busy as sources out of $(\kappa_i + \tau_i)$ busy access links. Φ can be calculated from the ratio of k and the number of destinations through the intermediate variable ξ:

$$\xi = \frac{k}{d} \qquad \text{(Eq. 43)}$$

If ξ is an integer, then the number of configurations is $\xi+1$. Otherwise $\Phi = \lceil \xi \rceil$. The number of access links busy as sources for a given configuration is chosen to be $\kappa_i = \Phi - i$. These result in $\kappa_i \cdot d$ busy network links. The remaining access links must be busy as destinations, so that $\tau_i = k - (\kappa_i \cdot d)$ because k is the total number of busy network links.

4.8.2.2 BLOCKING CORRECTION IN MAPPING EQUATIONS

The mapping equations above assumed no blocking. In order to be accurate, we now correct those equations to account for blocking.

1. Zero network links busy (k=0)

There is no blocking correction in this scenario as no network links at the first stage MCU are busy.

2. $1 \le k \le d - 1$

In this scenario, we made the assumption that if there are fewer than d network links busy, then the access links are not busy as sources. This ignores the fact that connection attempts can be site or network blocked and so it is possible for an access link to be busy as a source when $k<d$. For example, if $d=2$ and we want to calculate the probability of 1 network link being busy at a first stage MCU, then according to our assumption, there is only one configuration: one access link busy as a destination. But if we were to take blocking into account, there is one more configuration that is possible: one access link busy as a

source but with one of its destinations blocked. Another example is if *d=3* and we are interested in calculating the probability of 1 busy network link. One configuration is to have one access link busy as a destination. But considering blocking, another configuration is to have an access link busy as a source with 2 destination attempts being site or network blocked. The correction factor diminishes rapidly for configurations with more than one destination attempt blocked. We may justifiably assume that the blocking of a connection attempt is low (<5-10%) in any practical system of interest and therefore, configurations with higher order blocking terms have a negligible affect on the probability distribution of the network links. Configurations with first order blocking terms exist for the case where we want to calculate the probability of *d-1* network links being busy. The blocking correction factor χ_{d-1} for this case can be expressed as:

$$\chi_{d-1} = P_{ac}(1) \cdot p_s \cdot [P_s + P_n] \qquad \text{(Eq. 44)}$$

where χ_{d-1} is the probability that *(d-1)* network links are in use at a first stage MCU due to one access link being busy as a source, with one destination being blocked.

Therefore, the probability of *(d-1)* busy network links is:

$$P_b(d-1) = P_{ac}(d-1) \cdot (p_d)^{d-1} + \chi_{d-1} \qquad \text{(Eq. 45)}$$

3. $k = d$

As in the case of *(d-1)* network links being busy, we will now include a configuration with a first order blocking term. In this configuration, one access link is busy as a source with one destination attempt being blocked. Therefore, this access link uses *(d-1)* network links on the first stage MCU. The second access link is busy as a destination and is connected to one network link. A total of *d* network links are busied using *2* access links. The blocking correction factor is:

$$\chi_d = P_{ac}(2) \cdot 2 \cdot p_s \cdot p_d \cdot [P_s(d) + P_n(d)] \qquad \text{(Eq. 46)}$$

where $P_{ac}(2)$ is the probability of exactly 2 access links being busy, and there are two ways of choosing an access link out of the two that are busy. Therefore, the probability of d busy network links is:

$$P_b(d) = \left\{ \sum_{i=1}^{2} P_{ac}(\kappa_i + \tau_i) \cdot p_s^{\kappa_i} \cdot p_d^{\tau_i} \right\} + \chi_d \qquad \text{(Eq. 47)}$$

4. $d+1 \le k \le m$

If the number of network links that are busy is a multiple of the number of destinations or if k is one less than a multiple of d, we need to add a first order blocking correction factor. For example, if $d=2$ and we want to calculate the probability that 3 network links are busy, we could have 2 access links busy as sources with one destination being blocked for either one of the sources. On the other hand if we want to calculate the probability that 4 network links are in use, then 2 access links could be busy as sources with one destination blocked. A third access link would be busy as a destination.

If $k = \Lambda \cdot d$ *and* Λ is an integer >1, then the blocking correction factor is:

$$\chi_{(\Lambda \cdot d)} = P_{ac}(\Lambda + 1) \cdot \binom{\Lambda + 1}{1} \cdot p_s^{\Lambda} \cdot p_d \cdot [P_s + P_n] \qquad \text{(Eq. 48)}$$

If $k = (\Lambda \cdot d) - 1$ and Λ is an integer >1, the blocking correction factor can be generalized as:

$$\chi_{(\Lambda \cdot d)-1} = P_{ac}(\Lambda) \cdot p_s^{\Lambda} \cdot [P_s + P_n] \qquad \text{(Eq. 49)}$$

At this point we have derived equations for the distribution of the number of busy network links at a first stage MCU in the network. We need this to predict the carried traffic on the inter-MCU links, which will lead to P_n. The formulation involved mapping the independent states of the access links which follow the binomial distribution onto the correlated

states of the network links. The correlation arises from the nature of multicast calling. These mapping equations are also valid for point-to-point calls with $d=1$. The equations along with the blocking factors are implemented shown in Appendix A.

4.8.2.3 PROBABILITY OF NETWORK BLOCKED CONNECTION ATTEMPT, β

β is the second variable that we need to calculate in order to be able to predict the probability of network blocking, P_n, using Equation 35. It is the probability of any particular connection attempt being network blocked. Since we are interested in calculating the probability of any particular destination being blocked out of d, we draw a channel graph showing only one destination MCU (see Figure 4.9).

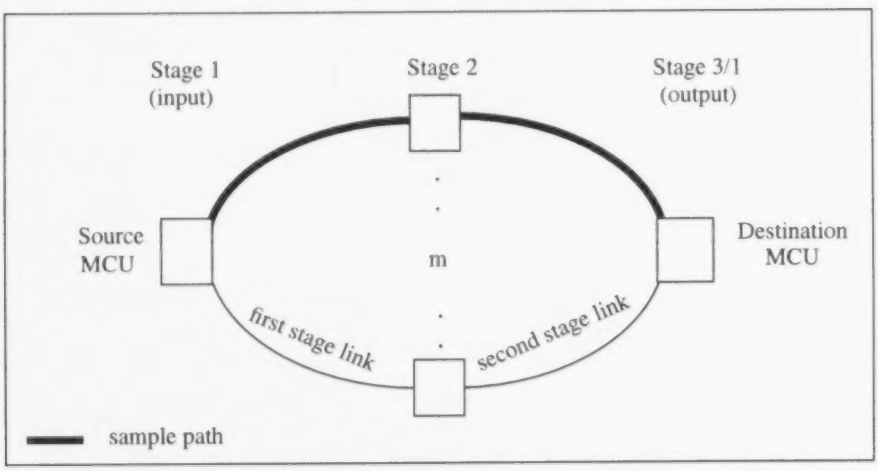


Figure 4.9. Channel graph for one source-destination pair

There are k first stage links free at the source MCU in the channel graph implying that $(m-k)$ are busy. A connection attempt will be network blocked if all second stage links which correspond to the first stage links that are free are busy. This implies that the number of busy second stage links, i, can range from k up to m and we want to calculate the probability that all k of the second stage links that correspond to the free first stage links are within a sample of size i. Therefore, the problem is similar to sampling without replacement and can be solved using the hypergeometric distribution:

$$\beta(k) = \sum_{i=k}^{m} P_b(i) \cdot \frac{\binom{k}{k} \cdot \binom{m-k}{i-k}}{\binom{m}{i}} \qquad \text{(Eq. 50)}$$

where $P_b(d, i)$ is the probability of exactly i second stage links being busy, and can be calculated using the mapping algorithm described in the previous section. There is only one way of choosing the k second stage links corresponding to the k free first stage links, and $\binom{m-k}{i-k}$ ways of choosing another $(i\text{-}k)$ busy second stage links out of the remaining $m\text{-}k$ second stage links. Dividing by $\binom{m}{i}$, the number of ways of choosing i busy second stage links out of m, gives the probability of this configuration.

4.8.3 SUMMARY OF NETWORK BLOCKING MODEL, P_n

This completes the analysis for the calculation of network blocking in the two stage switching network. We used an original model of a channel graph extended for multicast connections which involved the state probabilities of the network links that resulted in blocking. Unlike the access link state probabilities, the network link state probabilities cannot be approximated using the binomial formula due to the multicast bridging occurring at the first stage which causes correlated states on the network side MCU links. This was handled by recognizing the interdependence between the state probabilities of the access links which are independent and the network links which leads to the development of equations to map the state probabilities of the access links, calculated using the binomial formula, onto the state probabilities of the network links. At this stage P_n is calculable using the expressions above. Equations 35 through 50 are all subsidiary to Equation 34 which yields the probability of network blocking.

4.9 COMPUTATIONAL METHODOLOGY

We have now developed a system of equations to model the end-to-end blocking in the type of multicast network considered. In section 4.1 we defined end-to-end blocking as the probability that the network is unable to successfully complete any individual connection

attempt of a multicast call. This is $2 \cdot P_s + P_n$, where P_s is the probability of site congestion and P_n is the probability of network blocking.

A connection attempt is an attempt made to connect a source-destination pair which is a part of a multicast call with d destinations. The number of destinations per call is a random variable with distribution $P(d)$. Our analysis is general for $P(d)$ and therefore the equations for site and network blocking have been derived as functions of d. In the totally general case the average end-to-end blocking with d ranging from 1 to D is then:

$$P_{end} = \sum_{d=1}^{D} (2 \cdot P_s(d) + P_n(d)) \cdot P(d) \qquad \text{(Eq. 51)}$$

where $P(d) = 1/D$ for the specific case of a uniform distribution.

The practical issue is to calculate the probabilities of site and network blocking using the equations derived in this chapter Sections 4.2-4.7 for each value of d. The solution for P_s and P_n for a given d is necessarily iterative because the overall traffic intensity offered to each site, A_{tot}, is a function of P_s and P_n as given by Equation 17. The network and site blocking are therefore calculated jointly in conjunction with A_{tot}. In practice, starting values of $P_n=P_s=0$ and $A_{tot}=A_o(1+d)$ are used. Let E_s and E_n be the current estimates of $P_s(d)$ and $P_n(d)$ respectively. The following steps summarize the calculation of site and network blocking beginning with an estimate of each.

For a given d:

Step 1: *Let $E_s=E_n=0$*

Step 2: *Calculate the overall offered traffic, A, using the estimates.*

Step 3: *Calculate the site congestion, P_s.*

Step 4: *Calculate the access port utilization, a_c, using the estimates, E_s, E_n.*

Step 5: *Calculate probability of deflection, P_{def}, using the estimates, E_s, E_n.*

Step 6: *Calculate the probability of network blocking, P_n, using the estimates, E_s, E_n.*

Step 7: $\delta_s = |P_s - E_s|$ *and* $\delta_n = |P_n - E_n|$.

Step 8: *Set E_s=$P_s(d)$ and E_n=$P_n(d)$ and repeat* ***Steps 2 through*** *7 until δ_s and δ_n are negligible.* In our calculations and results that follow, the iteration is done until δ_s and δ_n are less than 10^{-5}.

Steps 1 *to* ***8*** *are repeated for all values of d.*

We expect the iteration to converge because the slope of the function at the fixed point, y=x is less than 1.

4.10 SUMMARY

In this chapter we have presented the analytical model used to evaluate the end-to-end blocking in a multicast network. The analysis interrelates traffic characteristics, particularly the originating offered traffic per source and the multicast factor d, to the network dimensions and results in a blocking estimate. The overall traffic intensity offered to each site is the sum of traffic originating at a site and the traffic for which the site acts as a destination. Site congestion involves the use of the Erlang B blocking formula for server groups and is inversely related to the number of access ports, p, at each site in the network and is directly related to the intensity of offered traffic. The access port utilization, a_c, and probability of deflection, P_{def}, were also derived. To calculate network blocking, we used an original model of a channel graph extended for multicast considerations which involved the state probabilities of the network links resulting in network blocking. The state probabilities of the network links are not binomially distributed due to the multicast effect at the first stage and were therefore calculated by mapping the access link states, onto the network link states. All the equations derived are general for the given distribution of the number of destinations per call, d.

5.0 MULTICAST NETWORK EMULATOR

In this chapter, we describe the network emulator that was used to experimentally test the theoretical model developed in Chapter 4. The emulator was to made to correspond as closely as possible to the actual functional operation of the network, as described in Chapter 3. The emulator is a discrete event simulator, coded in C.

5.1 EMULATOR INPUTS

The network topology file defines the number of sites, N, the number of access ports per site, p, and the number of first stage MCUs, X, and second stage MCUs, m. The user sites and all of the MCUs in the network topology are considered nodes in the network, interconnected via links. Each line in the network file (see Figure 5.1) defines a pair of nodes, the link interconnecting them and its state and the hop count. A link in the network has two states: busy and free. All the results presented here are based on links with unit length; this is equivalent to using hop counts, where a hop count of 1 implies that there is a one link path between a given pair of nodes in the network. Even if real-valued distances were used, the number of links used to make a connection would be the same, but the choice of links might differ.

span	nodea	nodeb	distance	working	spare
0	0	20	1	1	0
1	0	21	1	1	0
2	0	22	1	1	0
3	1	23	1	1	0
4	1	24	1	1	0
.	.	.	.	.	.

Figure 5.1 Sample Network File

In the analytical model, it was assumed that the originating traffic offered by each site in the network is A_o erlangs. This traffic intensity characterizes the ratio of the arrival rate λ and the departure rate μ of multicast calls originating from each site in the network. The mean arrival and departure rates are inputs to the emulator which allow testing a given network at various traffic intensities without modifying the structure of the network emulator. Note that A_o is the traffic originated by each site, rather than total offered traffic.

Other inputs include the seeds for the random number generator, the maximum number of call arrivals and the number of initial arrivals that are considered to define the duration of the start-up transient. The maximum number of destinations allowed, D, is also an input parameter.

5.2 CALL GENERATION

Each multicast call consists of a source site and d destinations chosen at random from the N sites in the network. The number of destinations is uniformly distributed ranging from 1 to D for each call request. No site is allowed to occur more than once in a call. The state of each call is stored in a binary min-heap [25] as an *event* for processing based on its arrival time. The binary min-heap stores the events as a binary tree with a time index such that the events are processed in increasing order of their time indices. Initially to load the network during a transient phase, we generate N calls with each of the N sites as a source and d destinations chosen randomly from the remaining $N-1$ sites using a uniform distribution. Each of these events is stored in the heap as a call arrival consisting of the source, destination sites and an arrival time.

The time interval to the next call arrival is generated from the mean interarrival time, $1/\lambda$, using the negative exponential distribution:

$$P(X \leq x) = F(x) = 1 - e^{-x/u} \qquad \text{(Eq. 52)}$$

where u is the mean value of the distribution. The standard approach to generating a value for a random variable is to first state that all values of $F(x)$ are equally likely. If the random

variable X has a distribution $F(x)$, then the values of $u=F(x)$ are uniformly distributed in [0,1]. The inverse mapping of $F(x)$ can then be used to generate a value for X. If x can be written as an inverse function derived from $F(x)$, then a random number from a uniform [0,1] distribution can serve as a choice for $F(x)$. For example, for the negative exponential distribution, the inverse map is:

$$x = -\ln(1 - F(x))/u \qquad \text{(Eq. 53)}$$

5.3 MULTICAST CALL ARRIVAL, PROCESSING AND DEPARTURE TIMES

The emulator processes one event at a time, taking the event at the current emulator time from the top of the binary min-heap. Both call arrivals and departures are stored in the heap. The emulator goes through a defined sequence of steps, depending on the event type.

5.3.1 NEXT CALL ARRIVAL

In the case of a call arrival, the first thing the emulator does is to generate the next arrival for the same source. The number of destinations in the next call is chosen randomly from a uniform distribution along with the actual target destination sites which are generated and stored at this time. The arrival time for the next event is generated from a negative exponential distribution.

The simulation time of the call arrival being processed is called *time_now*. The simulation time at which the newly scheduled arrival will be processed is (*time_now* + *inter-arrival time*), where the *inter-arrival time* is an instance of the random variable drawn from the negative exponential distribution.

5.3.2 CALL PROCESSING

After scheduling the next arrival, the emulator first checks to see if the source is congested. If all the access ports at the source are busy, the number of calls blocked due to the source being site blocked is incremented by one and the next event is processed.

If the source site has a free access port, the emulator makes the same check on each destination site. Each check made on the status of the access ports at a destination is recorded as an *attempt*. If the site is congested, the number of destinations blocked due to site congestion is incremented. If the destination site has a free access port, then the emulator enters the routing phase to attempt to find a network path between the source site and the destination site.

The first step in routing is to check for the possibility of deflection. If the source site and the destination site have access ports connecting to the same first stage MCU, the connection attempt is said to be *deflected*. In this case, the routing algorithm finds a path between the free access port at the destination site to the first stage MCU. The links along the path are set busy along with the access port at the destination site. The number of destinations connected is incremented for the call.

If the source and the destination are not connected to the same first stage MCU, the emulator attempts to find a path between the source MCU and the free destination access port through the switching network. If a path is not available, the connection attempt is said to be network blocked. If the attempt is successful, the two network links that make up the path are set busy along with the access port at the destination site, and the number of successful connection attempts in this call is incremented.

This is repeated for all the destinations one at a time. If the number of destinations sites successfully connected to the source is non-zero, a free path is sought between the source access port and the source MCU in the first stage, and the link on that path is set busy along with the access port at the source. Thus, we assume that any feasible portions of a multicast call will be connected. The pseudo-code for call processing is shown below:

```
If (source site blocked)
    (source_blocked)++
else
    For (all destinations in call)
        (site_attempts)++
        If (destination site blocked)
            (site_blocked)++
        else
            Find path between source and destination
            If (path not found)
                (network_blocked)++
            else
              set links in path busy
              set access port busy
              (site_connected)++
            end
        end
    end
end
```

5.3.2.1 ROUTING ALGORITHM

If a destination site is not congested and a deflected connection is not possible, an attempt is made to find a path between the source MCU and the destination access port through the switching network. Dijkstra's shortest path algorithm is used to find such a path [26]. Dijkstra's algorithm has $O(k\ n\ \log n)$ time complexity where n is the number of nodes in the network and k is the degree of each node.

5.3.3 CALL DEPARTURE SCHEDULING

After each destination in a call has been processed, if the number of destinations successfully connected is greater than 1, the *departure_time* for the entire multicast call is generated from the negative exponential distribution with the mean holding time, $1/\mu$. The actual simulation time at which the call departure will occur is (*time_now* + *inter-*

departure_time). The source site, along with the successfully connected destination sites and their respective paths are stored with the call departure event in the binary min-heap.

5.3.4 CALL DEPARTURE EVENT

When the event on top of the heap is a call departure, the emulator goes through a series of steps to free the network resources currently occupied by the call. The emulator traces each path and sets all the links on that path free so that they can be used by future call arrivals. The access ports at the destination sites and the source site are also set free. Note that the number of destinations that are in the call departure event may be less than d because of the fact that some destination sites may have been site or network blocked.

5.4 EMULATOR OUTPUTS

The primary role of the emulator is to validate the analytical model presented in Chapter 4 using a functional call-by-call simulation based on the given traffic conditions. The emulator gathers statistics by observing and recording the events in which we are interested. The values of blocking, deflection and link utilization that are observed by the emulator are averaged over the distribution of the number of destinations.

5.4.1 TRANSIENT EFFECTS

The emulator is started from an all-idle state. In the start-up phase, an originating call arrival time is generated for each site in the network. These call arrival events are stored in the binary min-heap based on their arrival times. The analytical model is based on the assumption of statistical equilibrium which implies that the probability per unit time of a call arrival is equal to the probability of a call departure. To obtain statistics of a similar nature using the emulator, the network is allowed to reach steady state before the emulator starts recording data. To ensure that the network has reached steady state, the emulator observes the time for which each network link is in use and updates the average time congestion of a link before an event is processed. For the results presented later 20,000 transient calls are processed before the emulator gathers statistics of interest. The transient for a test network is shown in the next chapter.

5.4.2 SITE BLOCKING

A connection attempt is said to be site blocked if all access ports are busy at the source or destination. A multicast call requesting connections to d destinations will make d connection attempts if the source is not congested. There are three ways of observing site blocking in the network emulator, two of which are event-based measures and the third of which is a time-based measure.

1) Site blocking at destination sites: The emulator observes the number of connection attempts that are blocked due to the unavailability of a free access port at destination sites over all calls. If all access ports at the source site are busy when the call arrives, no attempt is made to connect to the destination sites. Site blocking due to the unavailability of a free access port at the destination sites is expressed as the fraction of the total number of attempts made to connect to destination sites.

$$P_{s,1} = \frac{\sum destination\ blocked\ connection\ attempts}{\sum connection\ attempts} \qquad \text{(Eq. 54)}$$

2) Site blocking at source sites: This is observed by recording the number of originating *calls* blocked due to the source site having all access ports in use. In this context it is necessary to observe calls as opposed to individual connection attempts in each call because no connection attempts are made if the source is site blocked. The probability of blocking at a source site can be expressed as the fraction of all attempts made on sources.

$$P_{s,2} = \frac{\sum source\ blocked\ originating\ call\ attempts}{\sum originating\ call\ attempts\ made\ on\ sources} \qquad \text{(Eq. 55)}$$

3) Site time congestion: This is the time-based measure of site congestion that observes each site in the network and records the total fraction of time over which a site has all of its access ports in use. To obtain the site time congestion, the congestion inter-

vals are summed for all sites and then averaged over the number of sites and the total observed time, omitting the start-up transient:

$$P_{s,3} = \frac{\sum_{i=1}^{N} T_i}{N \cdot T} \qquad \text{(Eq. 56)}$$

where T is the total simulation time minus the start-up transient, and T_i is the total amount of time for which the i^{th} site is in the congested state. Each T_i is the sum of intervals, Δt_i's, during which site i has all its access ports in use.

In the analytical model, we made the assumption that the probability of site blocking can be calculated using the Erlang B formula and therefore call congestion (event based) must equal time congestion. This assumption can be validated with the emulator by seeing whether $P_{s,1}=P_{s,3}=P_s$.

5.4.3 ACCESS LINK UTILIZATION, a_c

Access port utilization can be defined as the average fraction of time for which an access port in the network is in use. The emulator observes each access port in the network and similarly records the total fraction of observation time for which it is in use. Access port utilization can be calculated as:

$$a_c = \frac{\sum_{i=1}^{N \cdot p} T_{p,i}}{(N \cdot p) \cdot T} \qquad \text{(Eq. 57)}$$

where $N \cdot p$ is the total number of access ports in the network and $T_{p,i}$ is the total time for which access port i is in use.

To verify the analytical calculation of the probability distribution of the access link states at a first stage MCU, we record the number of access links that are busy before processing an event. The number of observations is equal to the number of events proc-

essed, *numEvent*. If $U_{i,k}$ is the total number of times k access ports are busy at first stage MCU i, then the probability of k access links being busy at a first stage MCU is:

$$P(k) = \frac{\sum_{i=1}^{X} U_{i,k}}{numEvent \cdot X} \qquad \text{(Eq. 58)}$$

where X is the number of first stage MCUs.

5.4.4 DEFLECTION PROBABILITY, P_{def}

A connection is said to be deflected if the source site and the destination site have access ports connected to the same first stage MCU. Again, to use the emulator as an experimental check on the theory of Chapter 4, we record experimental deflection probability. When a call is processed and the source and destination are not site-congested, the emulator checks to see whether deflection is possible. Since we are interested in calculating the probability of a successful connection attempt being deflected, P_{def} can be calculated by taking the ratio of the number of deflected connections to the number of successfully connected attempts:

$$P_{def} = \frac{\sum deflected\ connection\ attempts}{\sum successful\ connection\ attempts} \qquad \text{(Eq. 59)}$$

5.4.5 NETWORK LINK UTILIZATION, a_n

This is the average fraction of the time when a network link in the inter-MCU switching network is in use. The procedure for calculating network link utilization is similar to that for access link utilization. The emulator observes each network link and records the time intervals over which it is in use. Network link utilization is calculated as:

$$a_n = \frac{\sum_{i=1}^{X \cdot m} T_{a,i}}{(X \cdot m) \cdot T} \qquad \text{(Eq. 60)}$$

where m is the number of network links at each first stage MCU so that $X \cdot m$ is the total number of network links between the first and second stage MCUs.

As in the case of access links, the probability distribution of network link states is obtained by recording the number of network links busy at each first stage MCU. If $W_{i,k}$ is the total number of times k network links at first stage *MCU i* are busy, then the probability of k network links being busy at a first stage MCU is:

$$P_b(k) = \frac{\sum_{i=1}^{X} W_{i,k}}{numEvent \cdot X} \qquad \text{(Eq. 61)}$$

5.4.6 NETWORK BLOCKING, P_n

If the routing algorithm is unable to find a path between the source MCU and the destination site, the connection attempt is said to be network-blocked. The emulator observes the number of connection attempts for which the routing algorithm is unable to find a path through the switching network. Network blocking is measured as the ratio of failed routing attempts to the total number of routing attempts made:

$$P_n = \frac{\sum failed\ routing\ attempts}{\sum routing\ attempts} \qquad \text{(Eq. 62)}$$

Attempts that result in a deflected connection are not included in this because a deflected connection does not go through the inter-MCU switching network.

5.5 TESTING & VALIDATION OF THE NETWORK EMULATOR

The emulator is a functional call-by-call emulation of a multicast network used for the performance evaluation of a test network and validation of the analytical model presented in chapter 4.

Given the role the emulator plays as the experimental validation for the theory in this work, an effort was made to verify the correct operation of each component and then the overall emulation.

The call generation component was verified for its correctness by ensuring that a call is initially generated for each site in the network. The number of destinations in each call is chosen randomly from a uniform distribution. For example, if the maximum number of destinations allowed per call is 4, then the probability of choosing a number between 1 and 4 must be 1/4. The same check is made to ensure that the destination sites are chosen at random and are uniformly distributed while including no destination twice in a call. The random number generator which is used in this emulator is provided as part of the UNIX ^{TM} operating system. The subroutine is called random() and produces a uniform random sequence in the range [0, 2^{31}-1]. It has a period of about 2^{35}. The generator can use from 8 to 256 bytes of state information; the multicast network emulator uses the maximum amount of state. A detailed check of the random number generator has been performed previously [25]. We generated 10,000 uniformly distributed numbers between 0 and 19 and found that the probability of each number being generated was 5%.

To check the accuracy of the call detail processing, we recorded individual call details for approximately 100 events including source and destination site details and the route set. Each event was analysed sequentially by hand to ensure the correctness of the call arrival, call processing and call departure algorithms. For a call arrival each route was traced through the network to ensure that the links in the routes did make the required connection. A check was made to see if the appropriate links and access ports were set to the busy state and that no busy link or access port was used before it was freed. Similarly, manual checks were made to ensure that the appropriate links and access ports were freed when a call departure occurred.

An important test performed on the overall emulator functionality was to validate Clos' condition for point-to-point non-blocking. Clos showed that for a two-stage single-sided network with n input ports on each first stage switch, the number of middle stage switches, m, required to have a non-blocking network for point-to-point traffic is:

$$m \geq 2 \cdot n - 1 \qquad \text{(Eq. 63)}$$

Therefore, we designed a test network which satisfied the above condition. The number of sites in the network, N, was 20. Each site was provided with one access port ($p=1$). The number of links on each first stage MCU for customer access, n, was chosen to be 4. This implied that a total of $X=5$ first stage MCUs were required. According to Clos' condition, a total of 7 second stage MCUs are required for the network to have zero network blocking for point-to-point calls. The emulation was run at an offered traffic intensity per site, A_o, of 0.2 Erlangs. In order to create point-to-point traffic only, all calls generated had one destination site. The emulation generated 80,000 point-to-point calls after equilibrium was reached. The results showed that the network blocking in the above network was indeed zero. This helped to validate the functionality and correctness of the emulator.

5.6 SUMMARY

An emulator programmed in the "C" language was used to experimentally examine the performance of multicast networks as described in the network model. It was designed from the point of view of trying to functionally replicate the call-by-call handling in a multicast network. Several aspects of the emulator's behaviour have been verified to ensure that it matches the assumptions and properties of the network model. The functional accuracy of the emulator was also verified by following individual event details and also by testing Clos's condition for point-to-point non-blocking networks. The emulator observes and records various events that are used to evaluate performance metrics including site congestion, network blocking, link utilization and deflection probability. The results are recorded only after the emulation has reached statistical equilibrium, in order to remove the non-stationary statistics in the start-up transient. Results obtained with the emulation are next used to validate the analytical results and the assumptions made in the analysis.

6.0 TEST RESULTS

In Chapters 4 and 5, we discussed the two methods that have been developed to analyse the performance of multicast networks based on the network model presented in chapter 3. The analytical model consists of closed form expressions to calculate the end-to-end blocking, whereas the emulator is a functional model of a multicast network and evaluates network performance by observing events. In this chapter, we present results obtained from the analytical model and the emulator for three hypothetical test networks. The emulator results are used as a benchmark to validate the analysis.

6.1 TEST NETWORKS

To test the analytical model of Chapter 4, three networks were designed without initial consideration for optimality in the sense that they were later optimized using the theory to be checked in this Chapter. The networks were chosen so as to test the analytical model and the emulator over a range of network dimensions and traffic intensities. The performance of each network was examined in detail using both the analytical model and the emulator.

6.1.1 NETWORK A

This test network consists of 20 sites with 3 access ports per site. The number of ports on each MCU for customer access is assumed to be 6, such that a total of 10 first stage MCUs are required, with two sites connecting to the same first stage MCU. The number of second stage MCUs is assumed to be 10 as well and the traffic intensity originating at each site is 0.2 erlangs. The maximum total number of ports available on each MCU is 16. Therefore, the dimensions of Network A (see Figure 6.1) are:

$$N=20,\ p=3,\ n=6,\ m=10,\ X=10,\ z=2,\ A_o=0.2\ erl,\ C=16$$

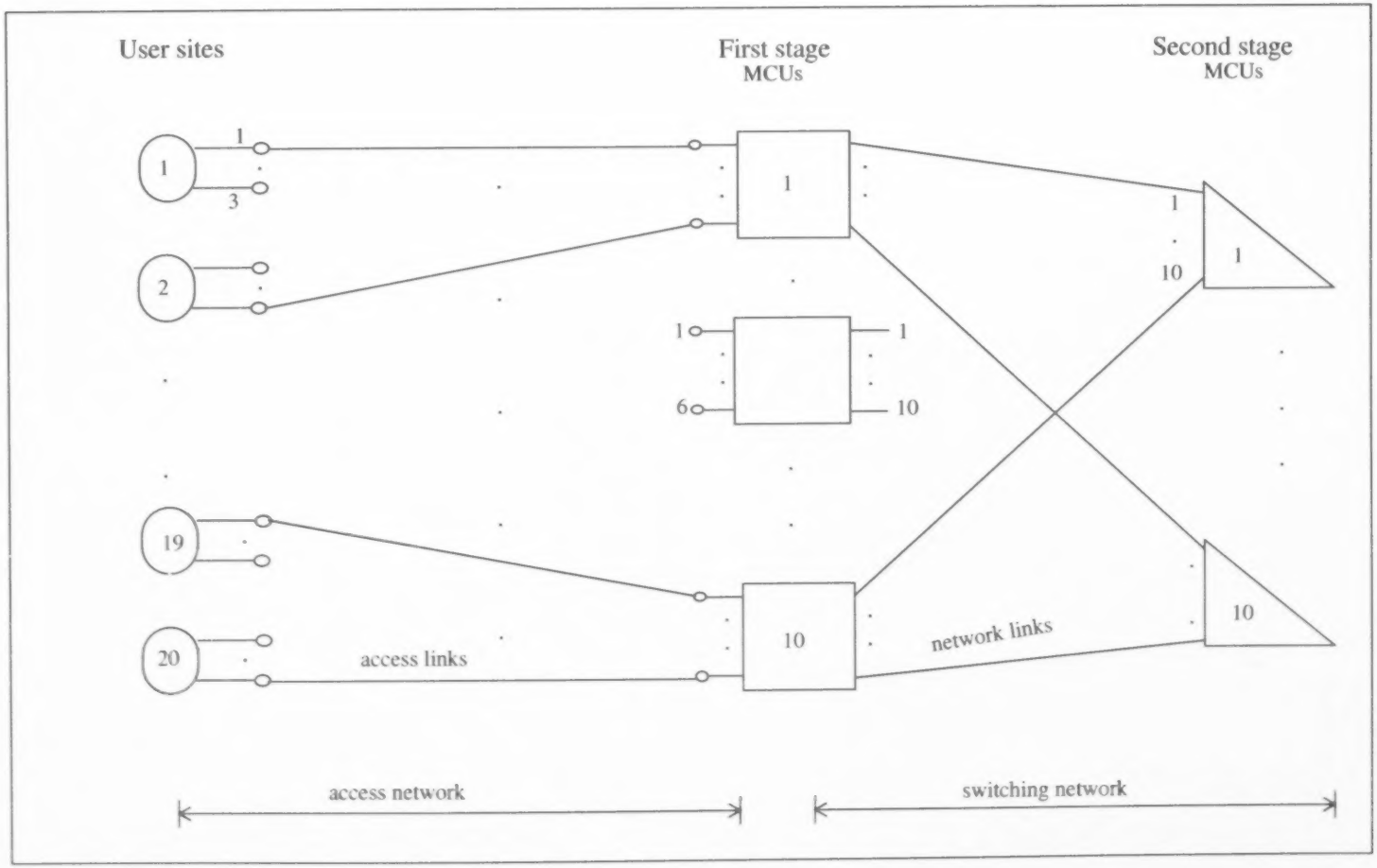


Figure 6.1 Network A

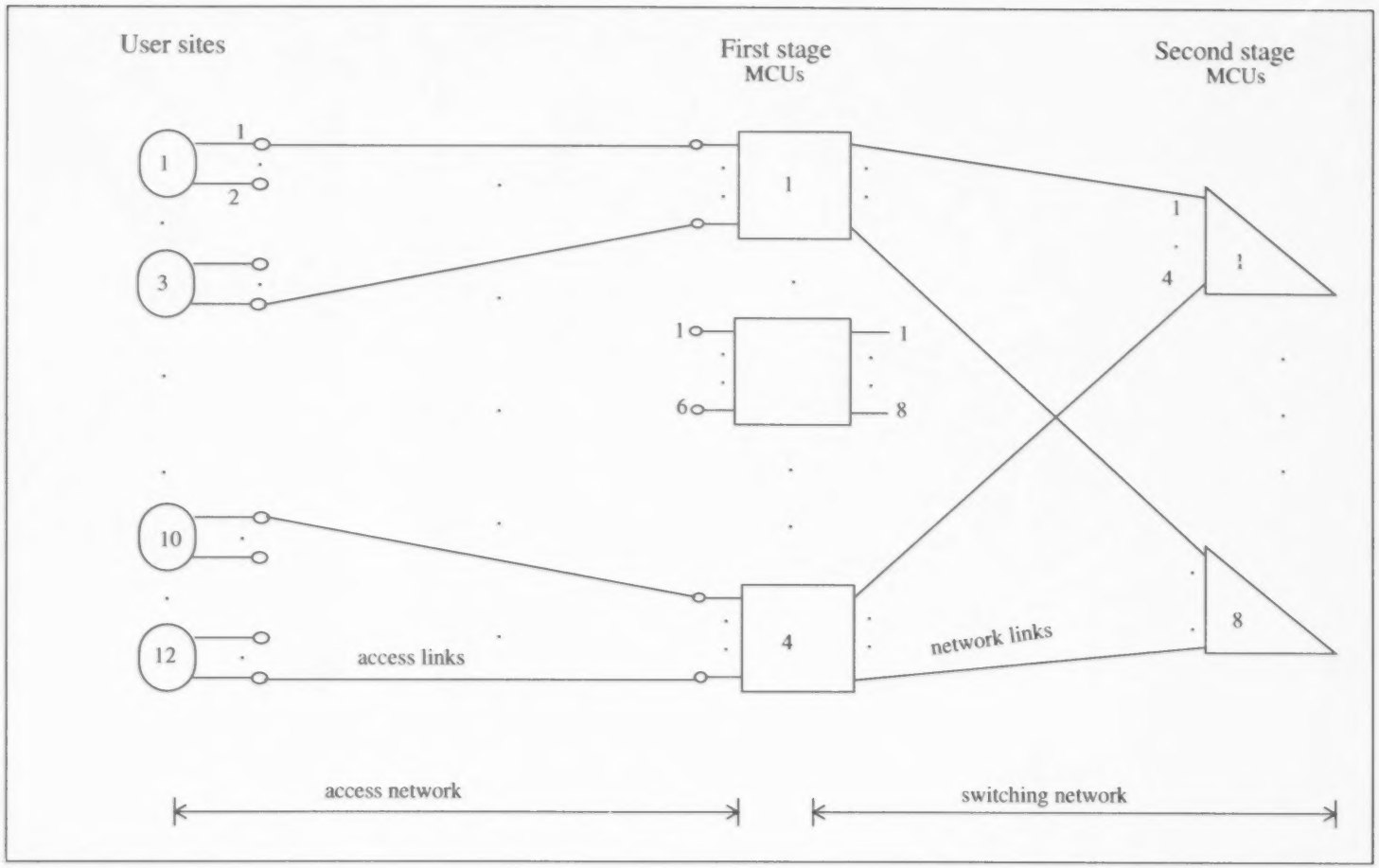


Figure 6.2 Network B

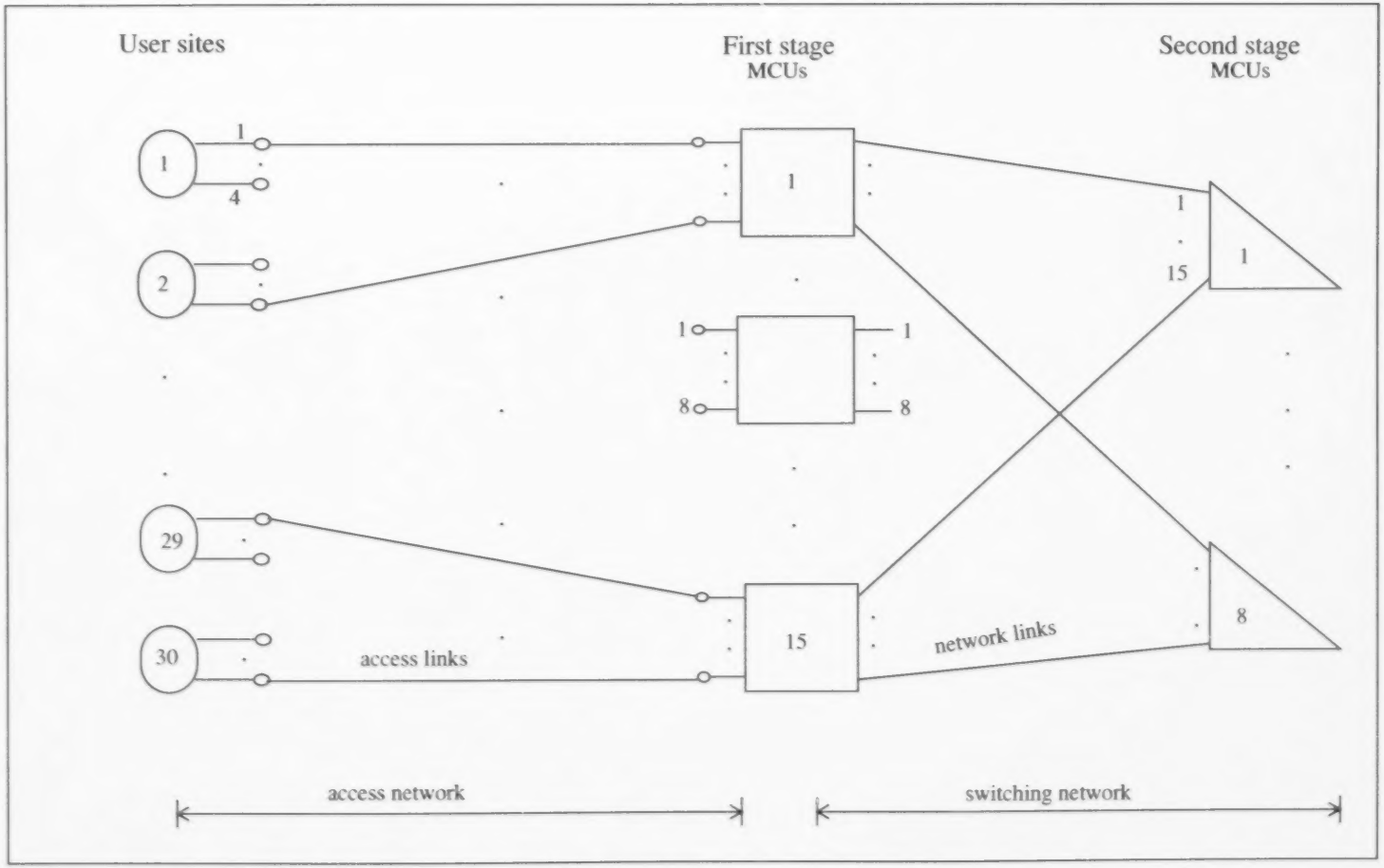


Figure 6.3 Network C

6.1.2 NETWORK B

In Network B, there are 12 sites, each with 2 access ports. There are 3 sites connecting to the same first stage MCU. The number of MCUs in the first stage is 4 with 6 ports for customer access and 8 ports for inter-MCU trunking. The traffic origination from each site is assumed to be 0.15 erlangs. The maximum total number of ports available on each MCU is 16. Therefore, Network B (see Figure 6.2) has the following dimensions:

$$N=12,\ p=2,\ n=6,\ m=8,\ X=4,\ z=3,\ A_o=0.15\ erl,\ C=16$$

6.1.3 NETWORK C

This network is larger than the other two. It serves a total of 30 sites each having 4 access ports with 2 sites connecting to the same first stage MCU. The number of ports for customer access on each first stage MCU is 8 and the number of network links is 8. The number of first stage MCUs is 15 and the traffic intensity originating from each site in the network is 0.35 erlangs. The maximum total number of ports available on each MCU is 16 (see Figure 6.3).

$$N=30,\ p=4,\ n=8,\ m=8,\ X=15,\ z=2,\ A_o=0.35\ erl,\ C=16$$

6.2 TESTING PROCEDURE

Each network was first analysed using the analytical model as it culminates in summary form and then the emulator. The testing procedure involved varying the maximum number of destinations allowed per call, D, and calculating the performance metrics. Although the aim is to be able to calculate end-to-end blocking, to diagnose and/or validate the analytical model we also compared theory versus experiment for each intermediate variable in the analysis for which there is a calculable counterpart in the emulator.

In the final test results with the emulator, a total of 80,000 calls were generated after statistical equilibrium was reached at 20,000 calls. Statistics were recorded after the start-up transient is discarded as explained in Chapter 5. The intrinsic variance of the system was dealt with by analysing results from multiple independent realizations with dif-

ferent values of the seeds for the random number generator. Using multiple independent runs made it possible to calculate confidence intervals for measured quantities. The results for each test network were obtained with a 95% confidence interval. The mean and standard deviation, σ, of results were obtained from the multiple runs and the 95% confidence interval is obtained as $mean \pm (2 \cdot \sigma)$. Since the emulator used a uniform distribution of number of destinations per call, all of the emulator results were automatically averaged over all possible values of *d*.

The analytical model presented in chapter 4 can be used to calculate the average end-to-end blocking, but the individual metrics including site blocking, access port and network link utilization, deflection and network blocking, are calculated as functions of *d*. In order to compare the analytical results and the emulator results, these metrics must be averaged over all values of *d*. For example, the probability of site congestion, $P_s(d)$, obtained from the analytical model is a function of *d* whereas the value from the emulator is an average value. The average site blocking is obtained by calculating site blocking for all possible values of *d* using the iterative steps outlined in chapter 4 and then averaging as:

$$P_s = \sum_{d=1}^{D} P_s(d) \cdot P(d) \qquad \text{(Eq. 64)}$$

where $P(d)$ is the probability distribution of *d* and is equal to $1/D$ for a uniform distribution. Other performance metrics are also averaged similarly.

6.3 START-UP TRANSIENT

Before any statistics are gathered by the network emulator, it was ensured that statistical equilibrium has been achieved. The emulator observes the network link utilization at every event that is processed along with the number of calls generated until that point. Figure 6.4 shows the start-up transient for Network A with $D=2$. The network link utilization reaches a steady state at a value of approximately 0.11 when the number of calls

already generated is approximately 15,000. The emulator starts gathering data for the performance metrics after 20,000 calls have been generated and generates 80,000 more calls.

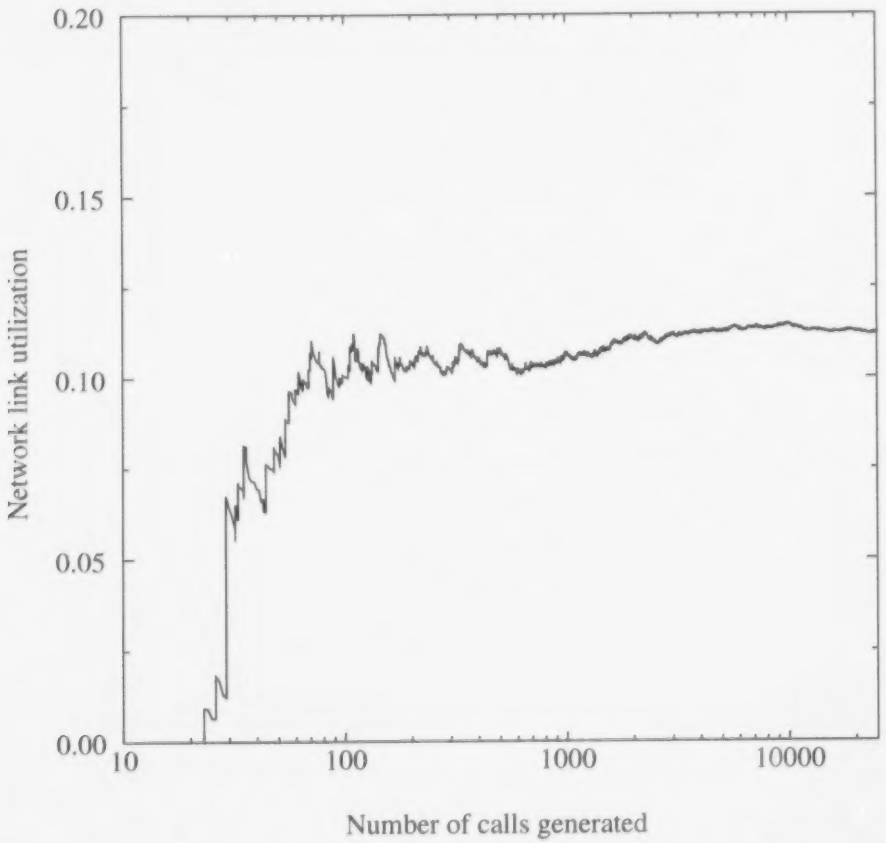


Figure 6.4. Start-up transient in Network A for *D=3*

6.4 EXPERIMENTAL VALIDATION OF EQUIVALENCE OF TIME CONGESTION and CALL CONGESTION IN THE ACCESS GROUPS

One of the assumptions made in the analytical model is that the probability of site blocking can be approximated using the Erlang B formula for a given number of access ports per site. A large number of potential users at each site is assumed in our model. If the mul-

ticast network is intended to serve an educational community, then the number of student and staff that are the potential users of the service are relatively large in number compared to the number of access ports. To validate this assumption, the emulator records site blocking as a time-based and call-based measure as explained in Section 5.4.

The results for all three networks are calculated and shown in Figures 6.5(a)-(c). They indicate that the time congestion is indeed very close to being equal to call congestion and therefore the assumption about site blocking being approximated by the Erlang B formula is valid. The remaining discrepancy could be due to the residual variance of a_n after statistical equilibrium is reached.

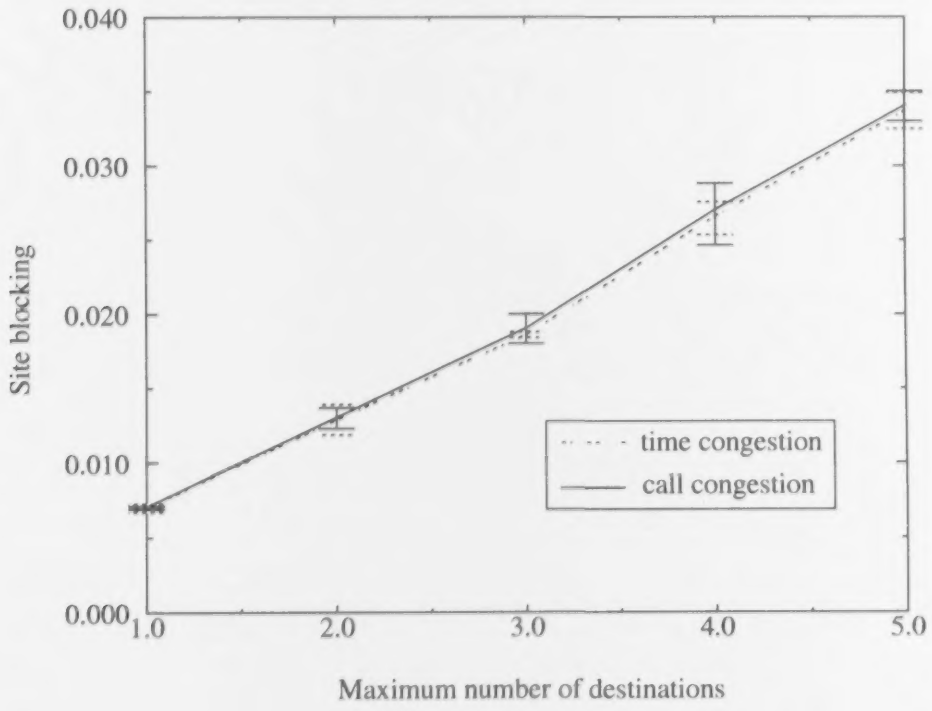


Figure 6.5(a). Time congestion vs. Call congestion for Network A

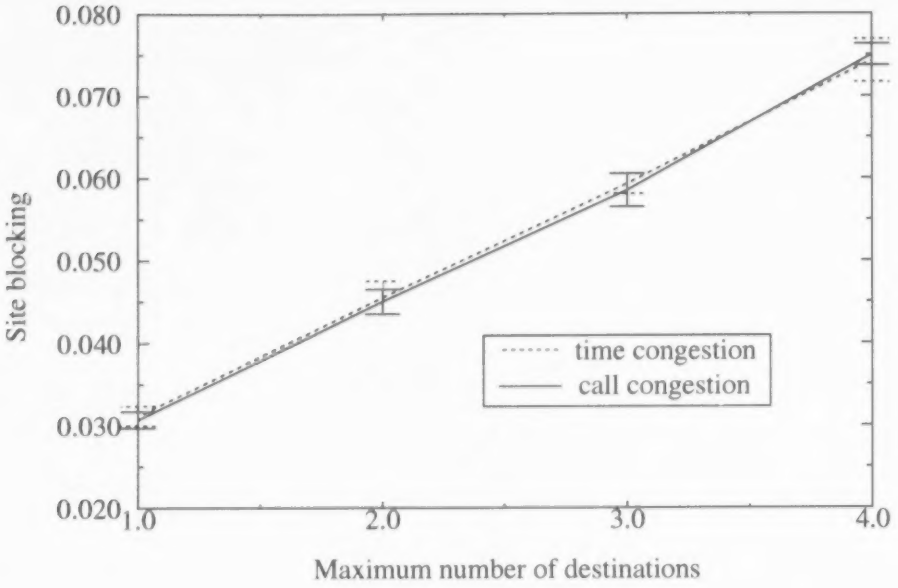


Figure 6.5(b). Time congestion vs. Call congestion for Network B

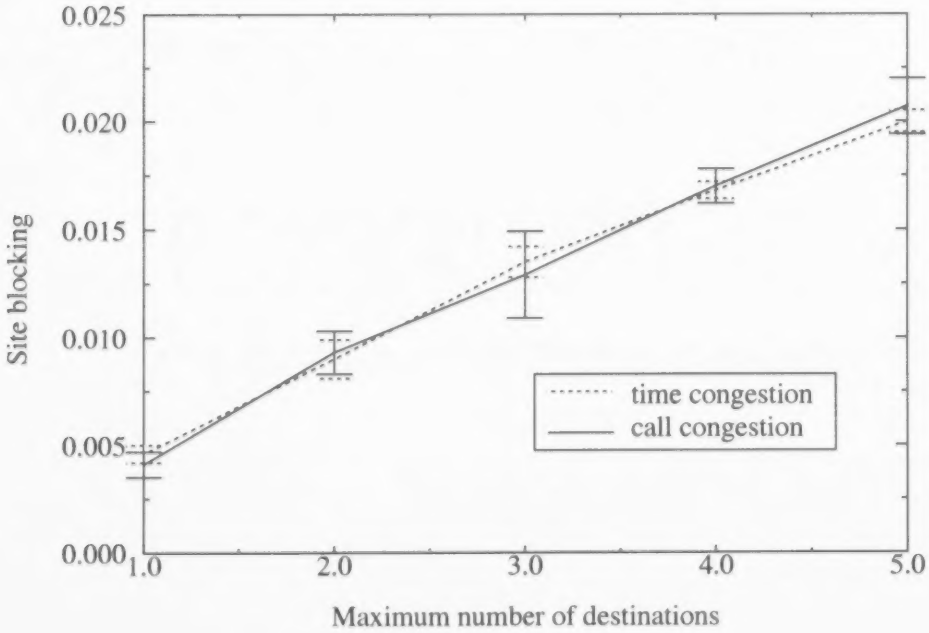


Figure 6.5(c). Call congestion vs. Time congestion for Network C

6.5 SITE CONGESTION/BLOCKING, P_s

Site congestion is the probability of time congestion in the group of access ports at a site, preventing the site from connecting to a source or from connecting to its destinations as part of a multicast call. Analytically, site congestion is calculated using the Erlang B formula for blocking in a group of servers (or in this case access ports), for a given amount of offered traffic. In the emulation, the fraction of time for which each site is in the congested state is recorded along with the number of connection attempts blocked due to site blocking.

For each network, the maximum number of destinations allowed per call is varied and the site blocking is calculated using the analytical model and the emulator. For all three networks, the emulator is a good approximation of the emulator results as shown in Figures 6.6(a)-(c). The probability of a connection attempt being blocked due to site blocking increases with D due to the fact that as the number of destinations requested per call increases, more network resources are utilized on a per call basis which increases the probability of a future connection attempt finding all access ports busy.

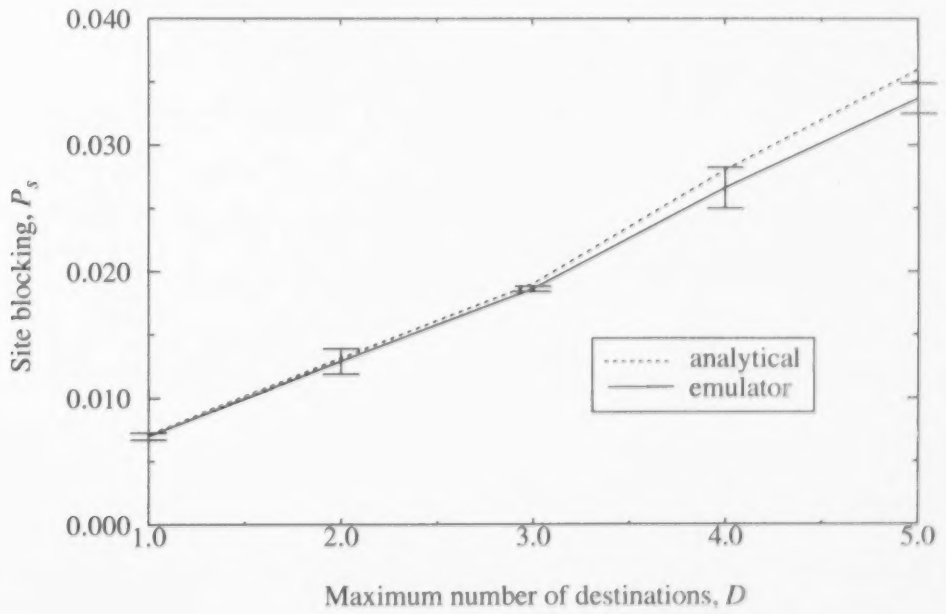


Figure 6.6(a). Site blocking in Network A

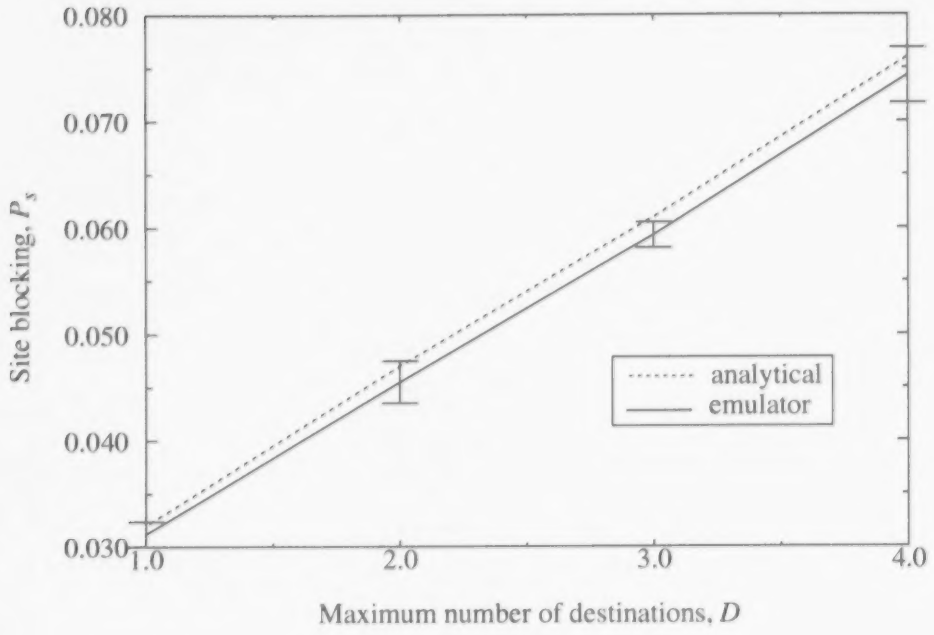


Figure 6.6(b). Site blocking in Network B

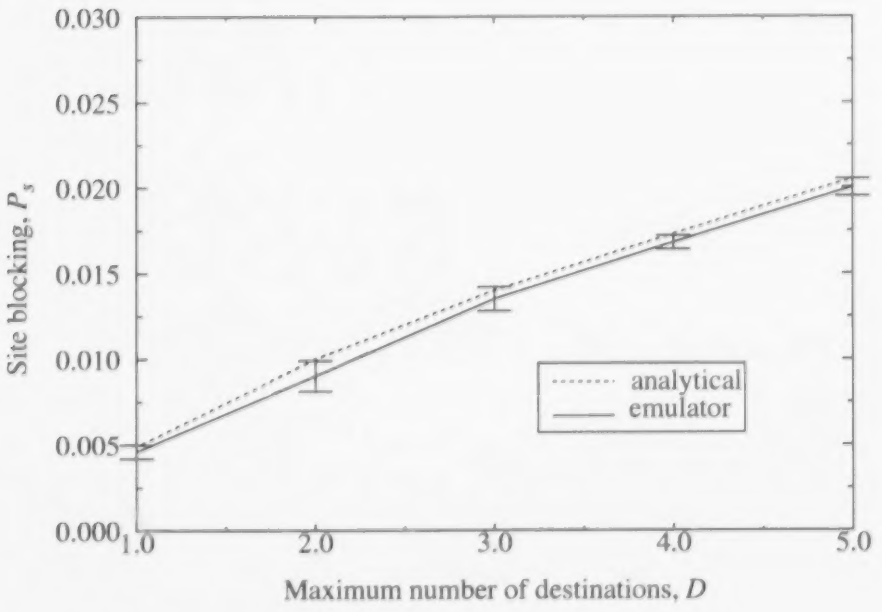


Figure 6.6(c). Site blocking in Network C

6.6 ACCESS PORT UTLIZATION, a_c

This is the average fraction of time an access port at a site is in use. The value of a_c has a direct correlation with site blocking in the network. Analytically, it is calculated by dividing the total traffic carried by the access ports at a site by the number of access ports. In the emulation, it is calculated by observing the state of each access link in the network and recording the intervals over which the link is in use. These are summed, averaged over all access ports, and then divided by the total simulation time minus the transient.

The emulator and analytical results for all three networks are in good agreement and are shown in Figures 6.7(a)-(c). The access port utilization increases with an increase in the maximum number of destinations allowed per call, D, where D is uniformly distributed. This can be attributed to the fact that more access ports are utilized per call as D increases.

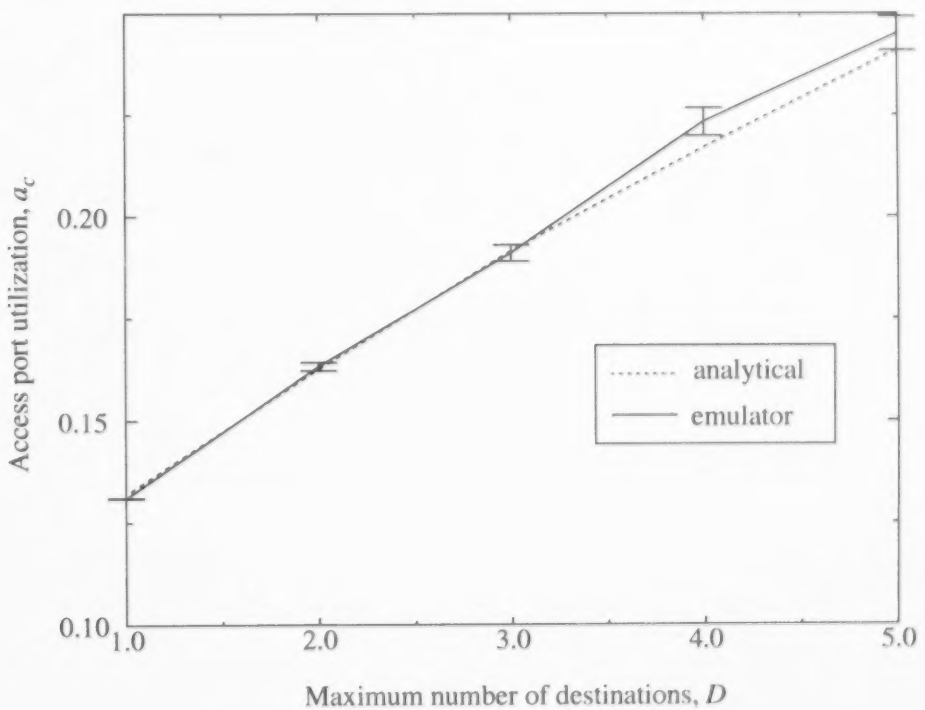


Figure 6.7(a). Access port utilization for Network A

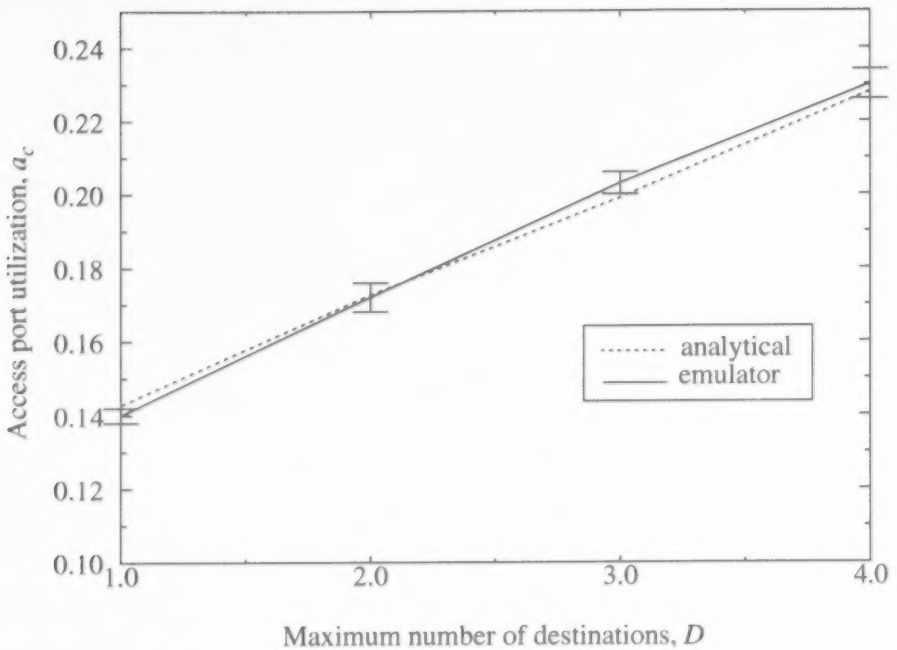


Figure 6.7(b). Access port utilization for Network B

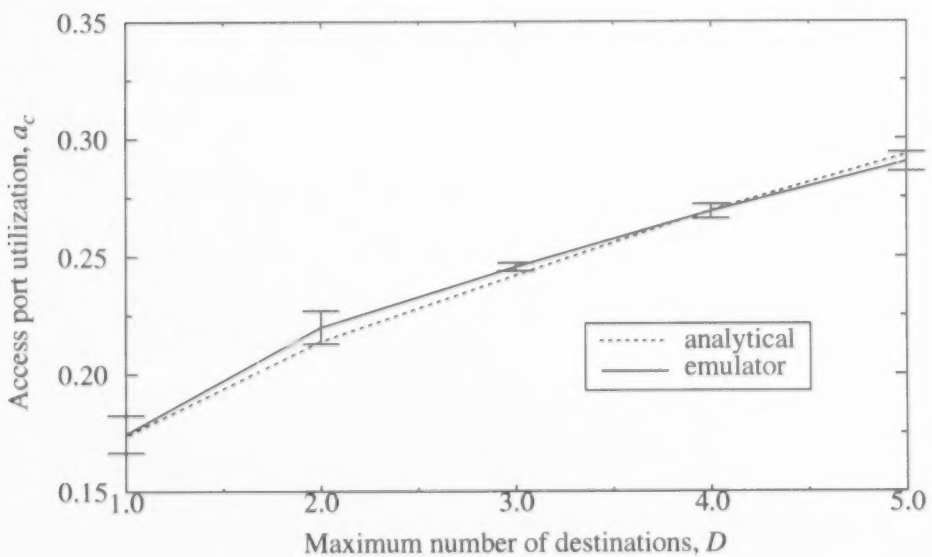


Figure 6.7(c). Access port utilization for Network C

In the analytical model, we make the assumption that the states of the access links at a first stage MCU are binomially distributed. Knowing the access port utilization averaged over all possible values of d, the probability that i out of n access links are busy at a first stage MCU is given by:

$$P'(i) = \binom{n}{i} \cdot (a_c)^i \cdot (1 - a_c)^{n-i} \qquad \text{(Eq. 65)}$$

To validate this assumption, the emulator samples the state of the access links at each first stage MCU before an event is processed. The emulator records the number of access links that are busy at each MCU and at the end of the run it calculates the fraction of the sampled events that found i links busy at each first stage MCU. This value is averaged over all first stage MCUs. The results are shown in Figure 6.8 for Network A assuming $D=3$, and indicate that the probability distribution obtained through emulation corresponds closely to the binomial distribution. This is a key assumption in determining the network blocking in the analytical model. Similar results were obtained for Networks B and C.

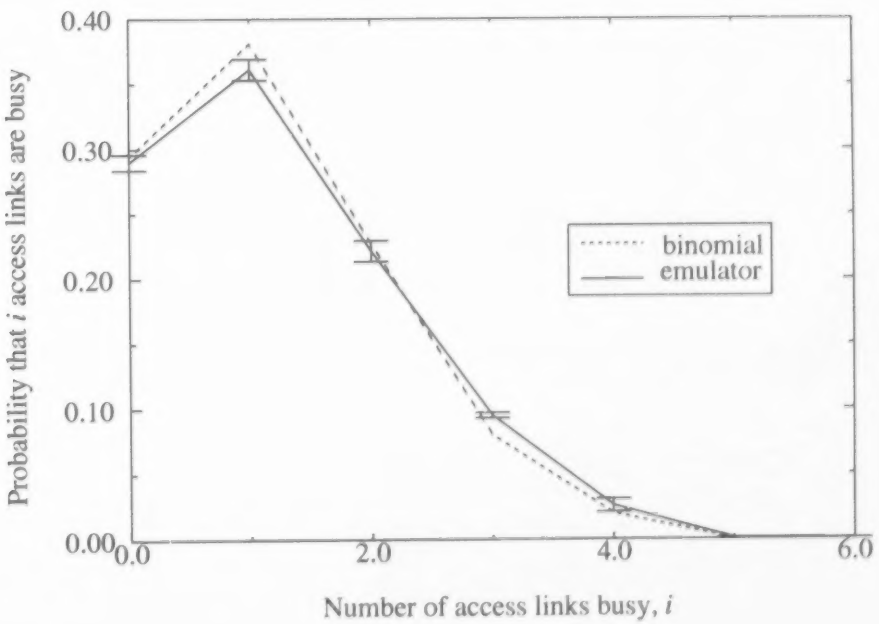


Figure 6.8. Binomial vs. Emulator: State probabilities of access links

6.7 PROBABILITY OF DEFLECTION, P_{def}

For a connection attempt to be deflected, the source site and the destination site must have access to the same first stage MCU. P_{def} is directly related to the number of sites in the network connecting to the same first stage MCU.

The analytical and emulator results are in good agreement, and are shown in Figures 6.9(a)-(c). The behaviour of the deflection probability is interesting with respect to D. It is dependant on the number of sites, z, in the network having access ports on the same first stage MCU. Therefore, once the source is chosen, a maximum of $z-1$ deflection possibilities remain for a multicast call. Therefore, P_{def} will increase until $z-1<d$, at which point we expect the probability of deflection to be more or less constant. For example, Figure 6.9(a) shows the deflection probability in Network A in which $z=2$. In this case, a maximum of 1 deflection can occur per multicast call irrespective of D, and thus P_{def} is more or less constant. The increase at higher values of D is due to the fact that blocking in the network is increasing. Looking back at Equation 23, P_{def} is calculated as the ratio of the expected number of deflections to the number of successful attempts. The number of actual successful connection attempts is reduced by network blocking at higher values of D and therefore we see a marginal increase in the probability of deflection. The same line of reasoning applies to Network C which also has two sites connecting to the same first stage MCU. In Network B which has $z=3$, the probability of deflection (Figure 6.9(b)) increases until $D=2$ because at most 2 deflections can occur per multicast call. At higher values of D, there is some increase due to the increase in network and site blocking which reduces the number of successful connection attempts.

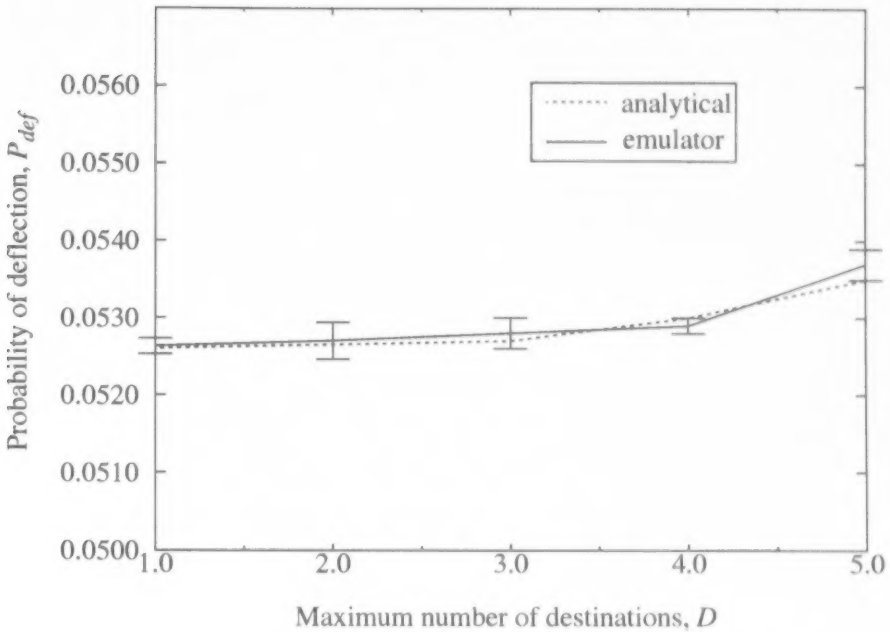


Figure 6.9(a). Deflection probability in Network A

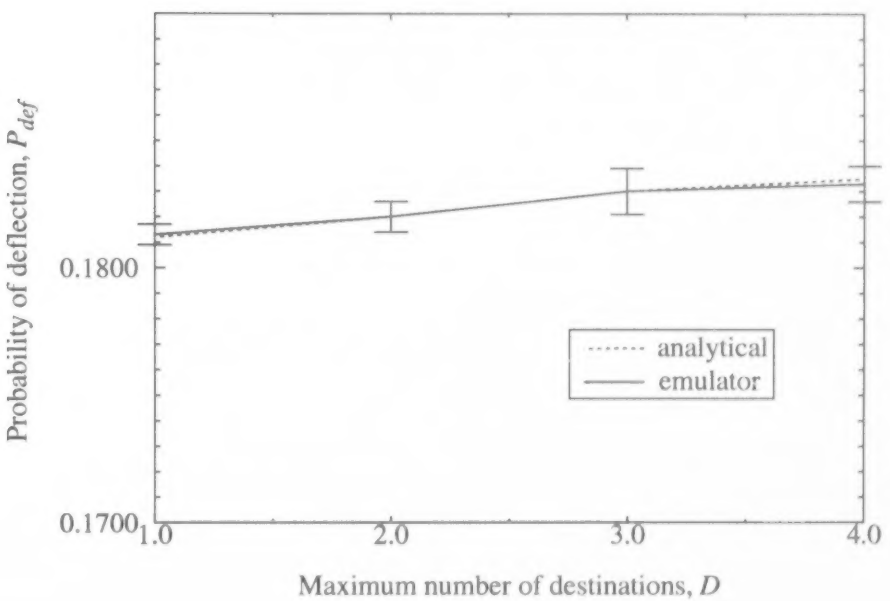


Figure 6.9(b). Deflection probability in Network B

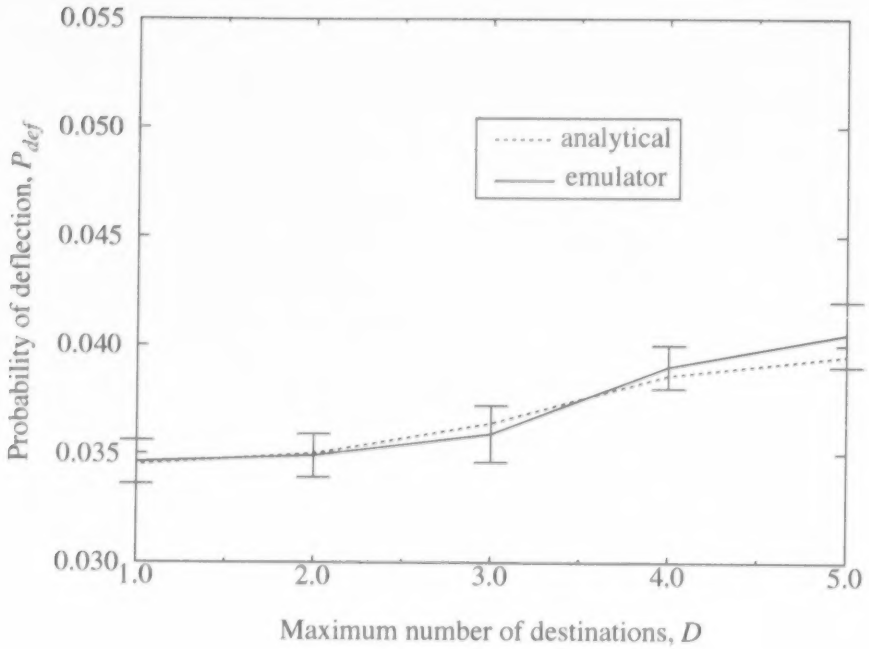


Figure 6.9(c). Deflection probability in Network C

6.8 NETWORK LINK UTILIZATION, a_n

This is the fraction of the time for which a network link is in use and is therefore its probability of blocking. In calculating a_n analytically, we must take into account the multicast bridging that occurs at the source MCU in the first stage. In the emulation, the state of each network link is observed and the intervals over which a link in use is recorded. At the end of the simulation, the intervals are summed, averaged over all network links and then divided by the simulation time after discarding the transient time.

The results obtained are shown in Figures 6.10(a)-(c) for each of the test networks and indicate that the analytical results agree with the emulation results. Network link utilization increases with D as more network links are utilized on a per call basis.

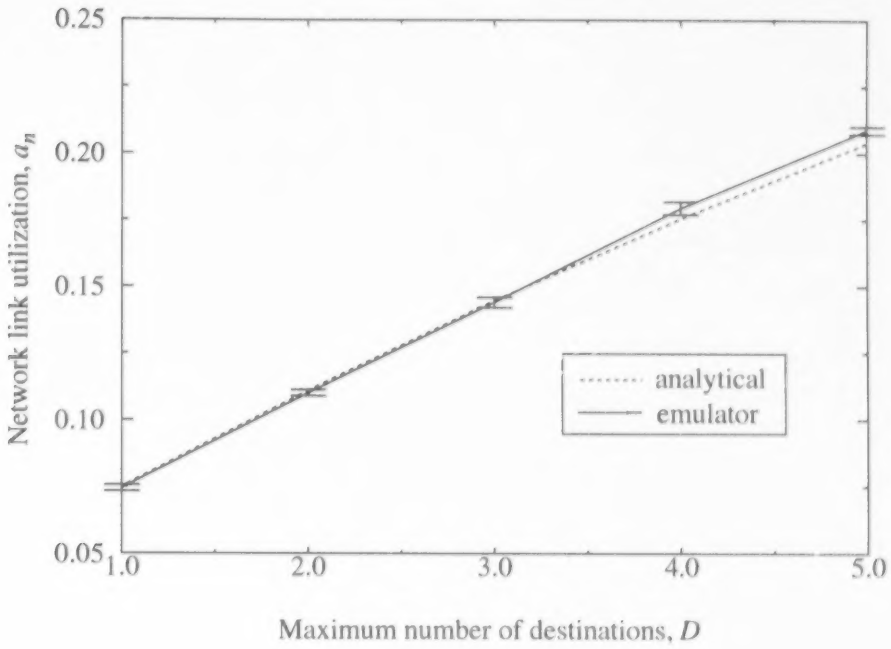


Figure 6.10(a). Network link utilization in Network A

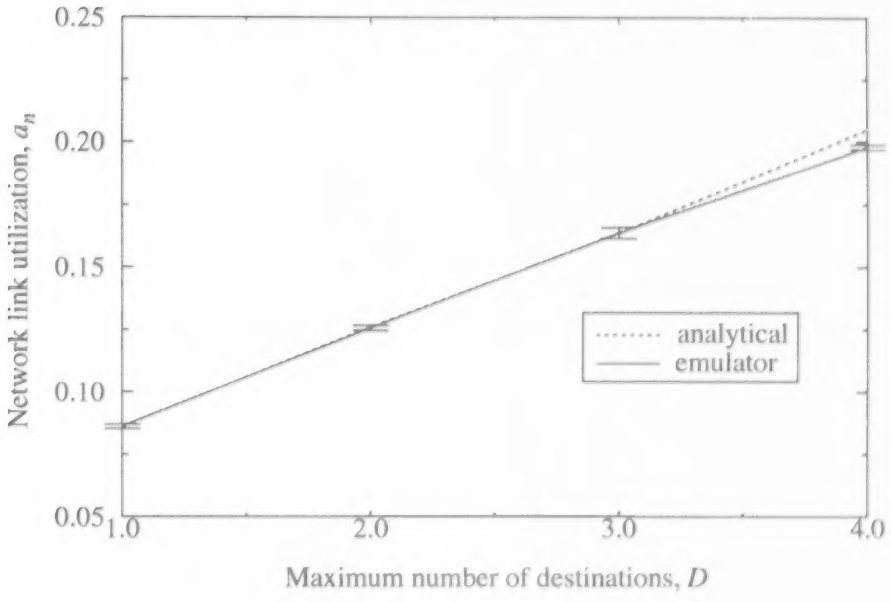


Figure 6.10(b). Network link utilization in Network B

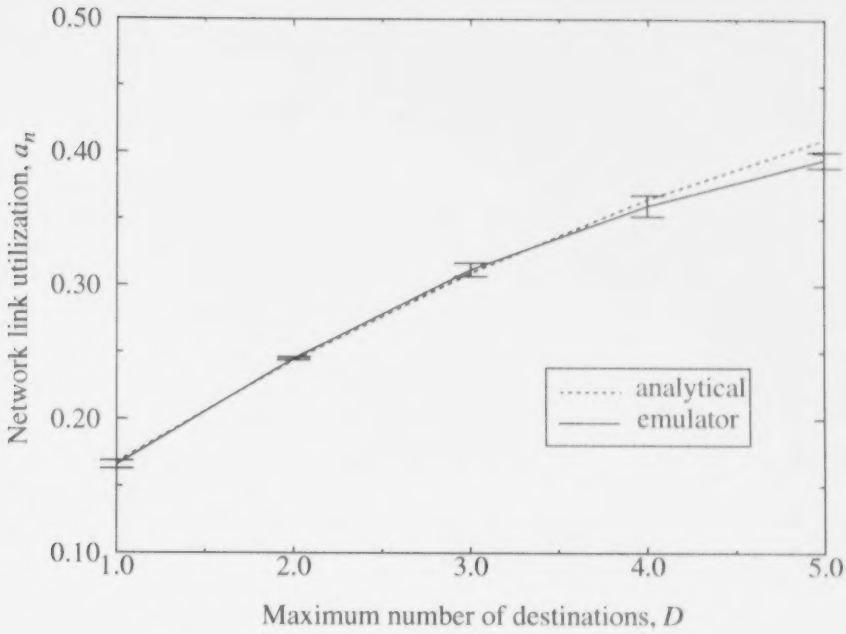


Figure 6.10(c). Network link utilization in Network C

In the analytical model we made the assumption that the states of network links are not binomially distributed due to the effects of multicasting. To check this assumption, we calculate the probability that i of m network links are busy at a first stage MCU:

$$P''(i) = \binom{m}{i} \cdot (a_c)^i \cdot (1 - a_c)^{m-i} \qquad \text{(Eq. 66)}$$

Figure 6.11 shows that there is a significant difference between the observed state distribution and that predicted by the binomial distribution. Therefore, we developed a method to map the access link states onto the network link states, where the access link state distribution is obtained through the binomial formula. The procedure to calculate the distribution via emulation is similar to that for access link distribution. The emulator observes the state of network links at each first stage MCU for each sample event and records the number of network links busy at each MCU. At the end of the run the emulator calculates the fraction of the sampled events that found i network links busy at each first stage MCU. This is averaged over all first stage MCUs. The results are shown in Figure

6.11 for Network A assuming $D=3$. The result obtained through the mapping method is a good approximation of the emulation result. Similar results were also obtained for the other two networks.

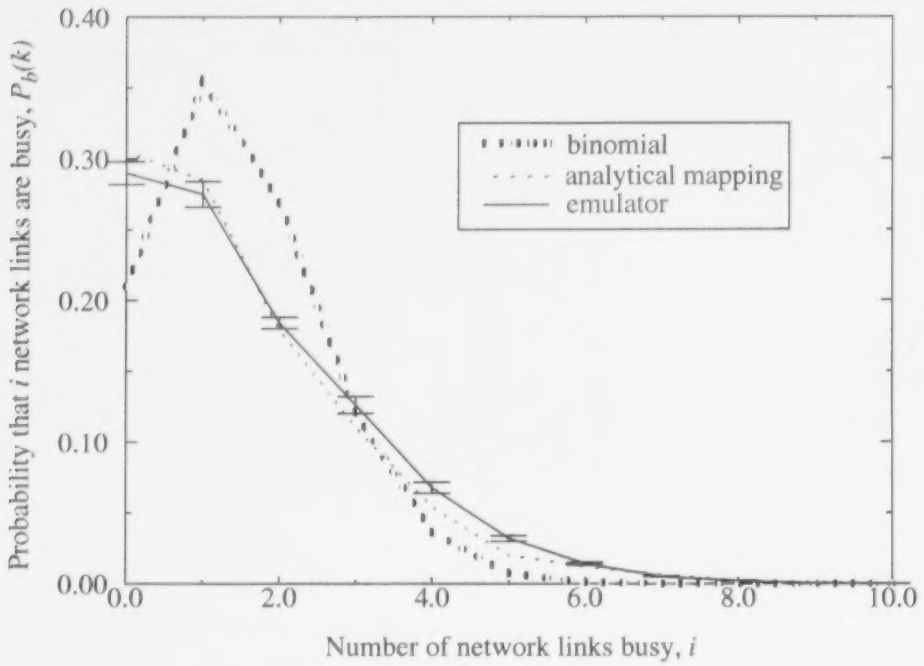


Figure 6.11. State probabilities of network links

6.9 NETWORK BLOCKING, P_n

A connection attempt is said to be network blocked if it cannot be completed due to the unavailability of one or more network links that are required to successfully complete the requested connection. Analytically, calculation of network blocking involves the use of a model of a multicast channel graph developed in this thesis. The analysis involves the use of the probability distribution of the network links to determine the states that cause network blocking. In the emulator, network blocking is calculated by taking the fraction of the connection attempts on the switching network that are blocked. Deflected connections are not attempts on the switching network.

Figures 6.12(a)-(c) show that the analytical prediction is a good approximation of the emulator results for all of the test networks. As expected, P_n increases with D due to the fact that more network links are used to satisfy connection requests when the maximum number of allowed destinations is increased.

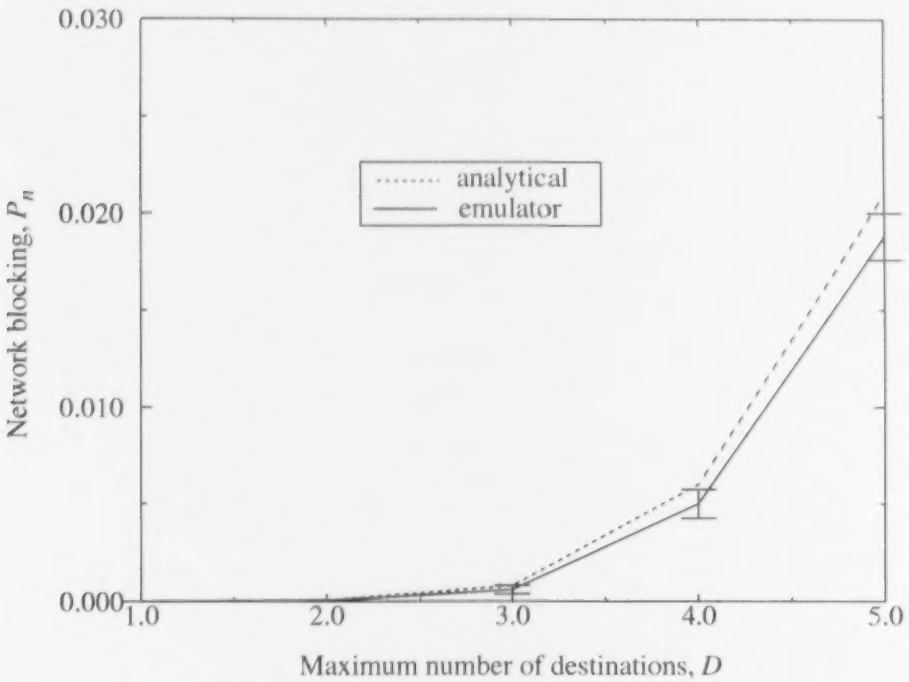


Figure 6.12(a). Network blocking in Network A

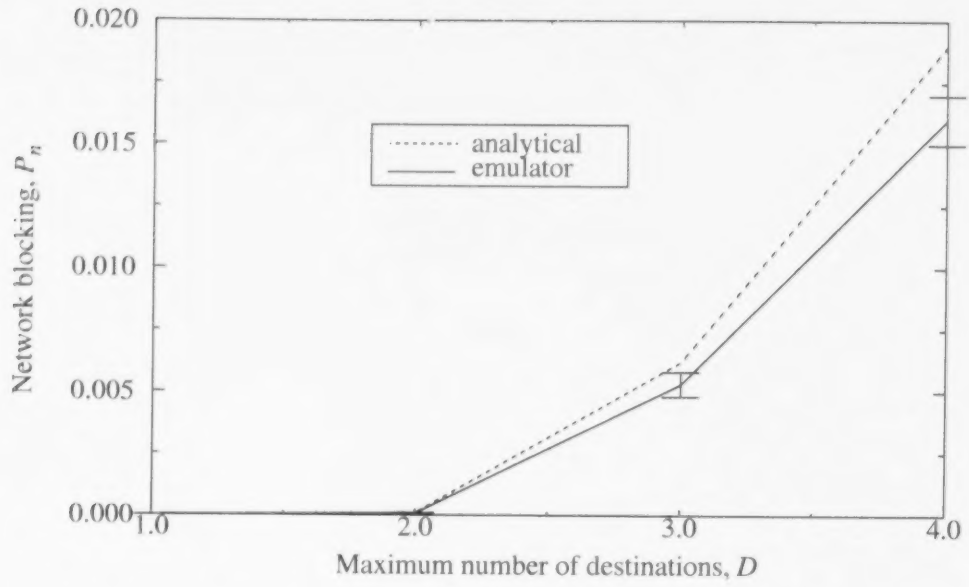


Figure 6.12(b). Network blocking in Network B

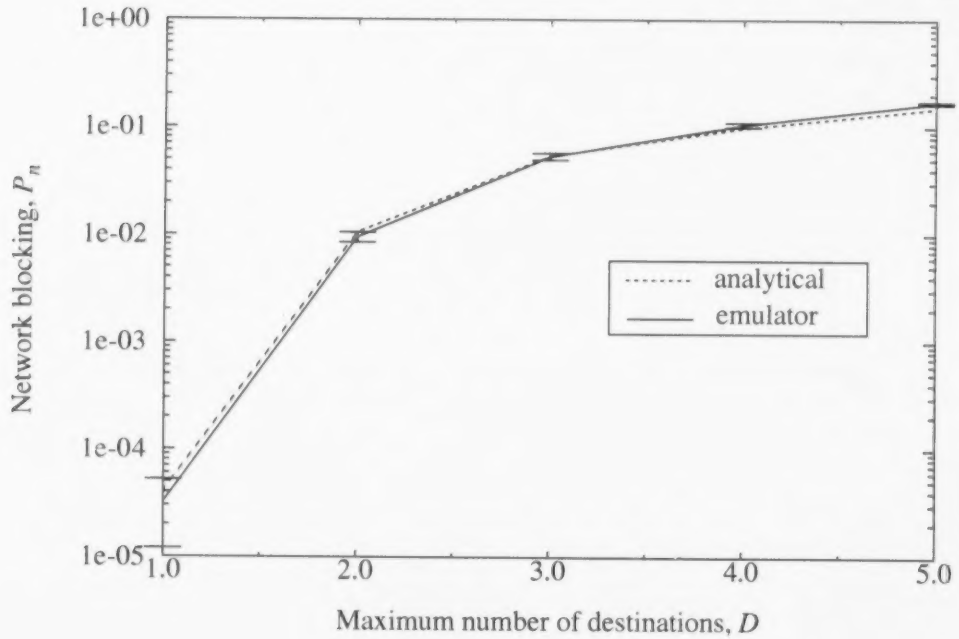


Figure 6.12(c). Network blocking in Network C

6.10 END-TO-END BLOCKING, P_{end}

The end-to-end blocking from a connection attempt point of view is a combination of site and network blocking. Both P_s and P_n as predicted analytically have been shown to be in good agreement with the emulator. Therefore, we expect to the end-to-end blocking results to do the same.

Figures 6.13(a)-(c) illustrate the analytical and emulator results for the three test networks. The analytical results are a good approximation of the emulator for low blocking levels (<10% blocking). The blocking increases as the number of destinations allowed per call increases.

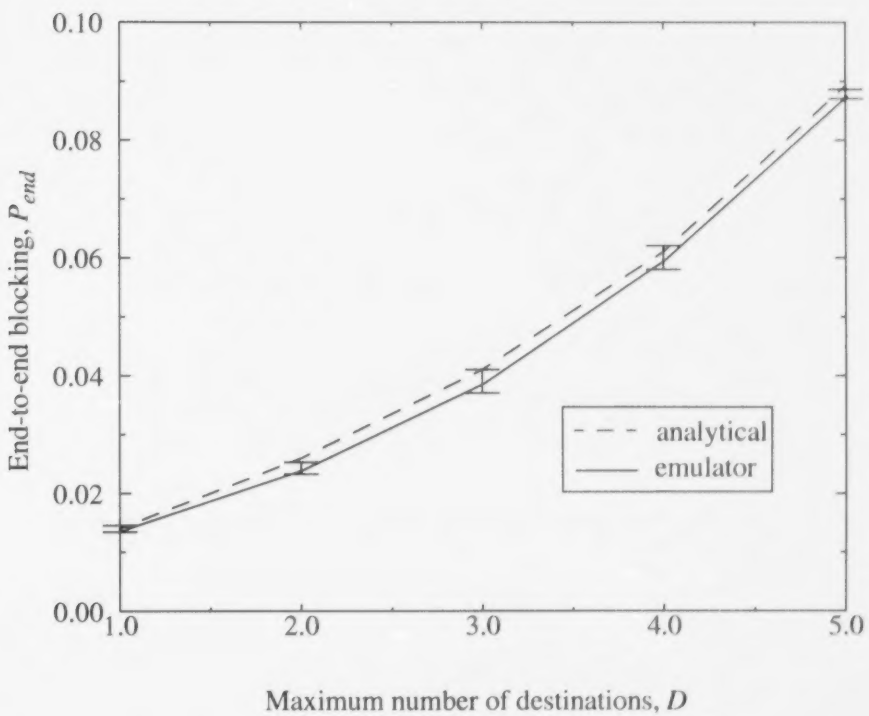


Figure 6.13(a): End-to-end blocking in Network A

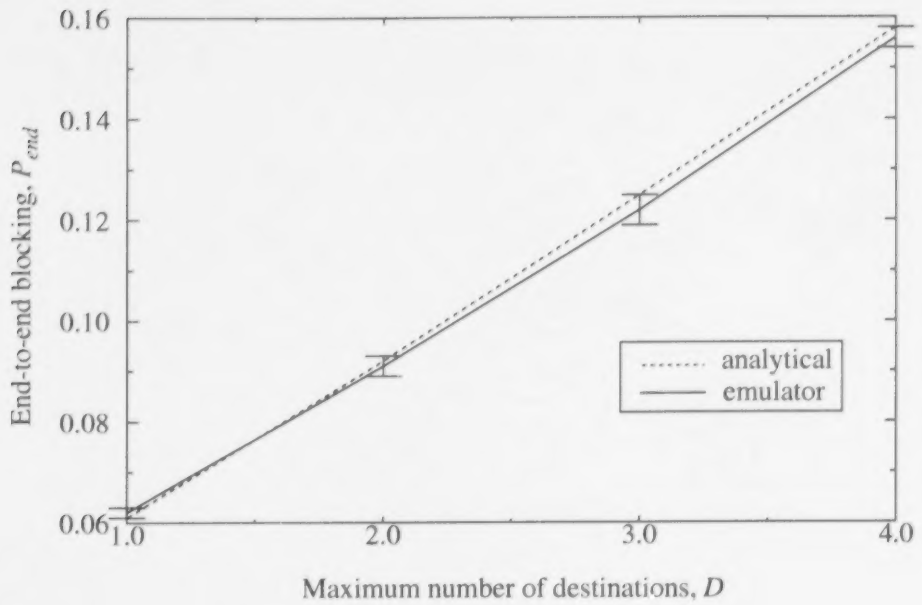


Figure 6.13(b). End-to-end blocking in Network B

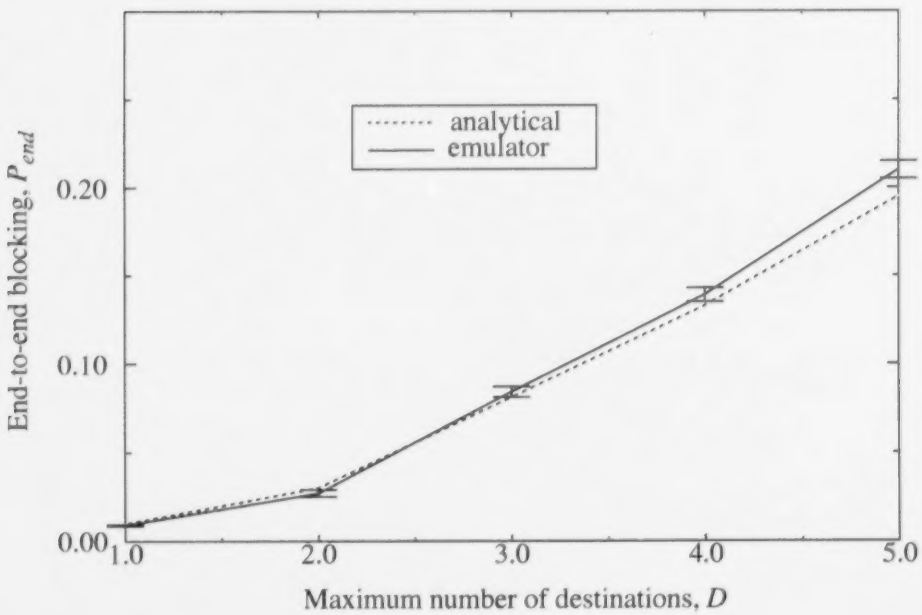


Figure 6.13(c). End-to-end blocking in Network C

6.11 SUMMARY

This chapter reports validation of the analytical model using the emulator results as an experimental benchmark. The analytical model and the emulator are compared using three networks with varying dimensions and traffic intensities. All intermediate variables derived analytically, including site and network blocking, were compared to the emulation results and found to be in satisfactory agreement. Any discrepancies tend to increase at higher values of D because as the maximum number of destinations per call increases, blocking in the network model becomes significant. This introduces higher-order effects in the system that have only been approximated analytically. But at the grade of service of 5-10% blocking or below at which a multicast service of the nature described in this thesis is most likely to operate, the analytical model is a good approximation of the emulator.

7.0 MULTICAST NETWORK DESIGN OPTIMIZATION

In Chapter 3, the network model that serves as a basis for the design of multicast networks considered as a part of this research work was described. The analytical and emulator results were compared in Chapter 6 and indicated that the analytical results for all performance metrics are a good approximation of the emulator results. In this chapter we therefore proceed to use the analytical model as the basis for an optimization technique to design a network that achieves a given grade of service at minimum cost.

7.1 MOTIVATION

To understand the opportunity to optimize the network, we need to understand two basic blocking mechanisms in the network model. Site congestion, P_s, is the probability of a connection attempt being blocked due to the unavailability of a free access port at the site in question. If the amount of traffic offered to a group of access ports is fixed, site blocking is inversely related to the number of access ports. On the other hand, network blocking, P_n, occurs between the first and second stage MCUs and, is inversely related to the number of second stage MCUs, m. An increase in m reduces the probability of a connection attempt being blocked through the switching network. This is due to the availability of more parallel paths from the source MCU to the destination MCU.

The number of access ports at each site and the number of second stage MCUs are dependant on how the ports on each first stage MCU are partitioned because the total number of ports on an MCU is fixed. The ports on each first stage MCU must, in this architecture, be partitioned to provide n user access ports and m inter-MCU junctoring ports such that the total number of ports is $n+m=C$, the total number of MCU ports. If the number of access ports is increased to lower site congestion, there will be fewer ports available for inter-MCU junctoring, thus increasing network blocking. Thus the first-stage MCU split in port allocations needs to be "optimally" determined.

The prime reason for the optimization opportunity is that site and network blocking show opposite behaviour as the number of ports for customer access on each first stage

MCU is varied. This is illustrated in Figure 7.1 for test specifications of $N=20$, $p=2$, $D=3$ and $C=16$ and $X_{max}=20$. Each site originates traffic of 0.1 erlangs. For each value of X_{max}, the value of n is varied. Site blocking decreases as n increases due to the fact that the number of access ports at each site is increasing. On the other hand, network blocking increases with n as the number of network links, m, decreases because the number of ports per MCU, C, is fixed.

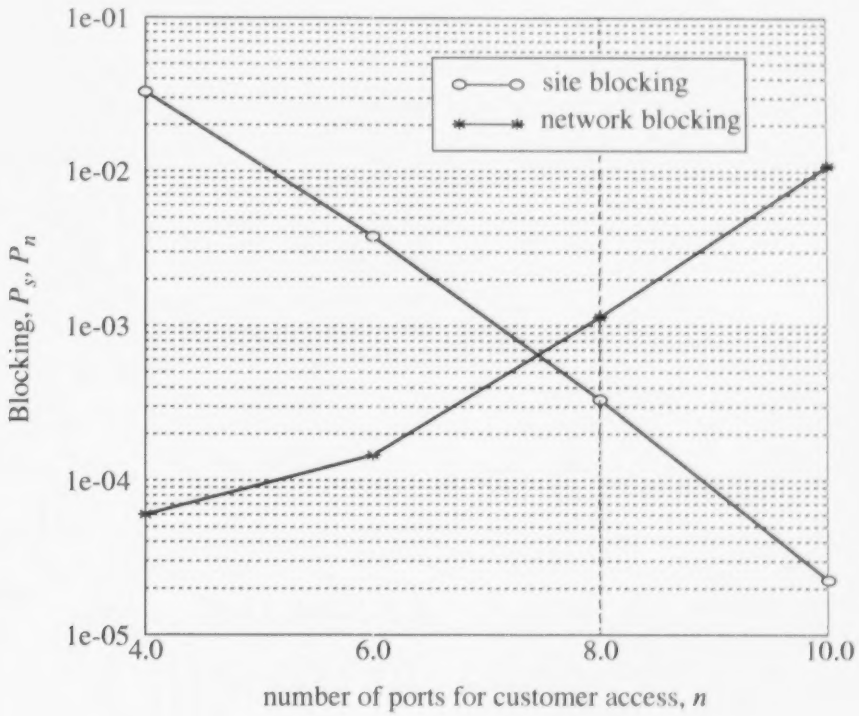


Figure 7.1. Site blocking vs. Network Blocking for varying n

Therefore, low site congestion and low network blocking are in contradiction with each other, and are directly affected by the split of the ports at each first stage MCU. Thus, there is an optimum split of the access ports at each first stage MCU such that the end-to-end blocking, $P_{end} = 2 \cdot P_s + P_n$, is a minimum. In summary, the problem may be stated as follows: *Find the smallest total number of MCUs, and their allocation as first and second stage switches, plus first stage port partitioning between user access and inter-MCU*

junctoring so that target end-to-end blocking or individual site and network blocking objectives are satisfied.

7.2 OPTIMIZATION STRATEGY

The problem of optimizing MCU configuration involves inter-relating various parameters in the network topology. Looking at a first stage MCU in Figure 7.2, the end-to-end blocking in the multicast network is dependant on how the ports at the first stage MCU are split between customer access, n, and inter-MCU trunking, m.

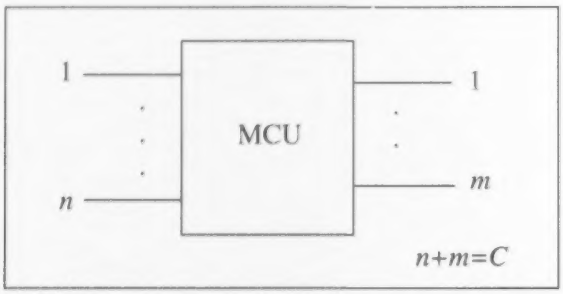


Figure 7.2. Partitioning of first stage MCU

If the total number of ports on each MCU is C, then the following relation is true:

$$m \leq C - n \qquad \text{(Eq. 67)}$$

In the network model we assume that the multicast network topology (Figure 7.3) is symmetric and regular. All N sites have the same number of access ports, p; all first stage MCUs are of size (n x m), all MCUs have the same total number of ports, C, and there is exactly one link between each pair of first and second stage MCUs. Therefore, there are m second stage MCUs each having X ports in use. Note that to achieve greater savings, we only require the second stage MCUs to have X ports rather than C, where X is the number of MCUs in the first stage.

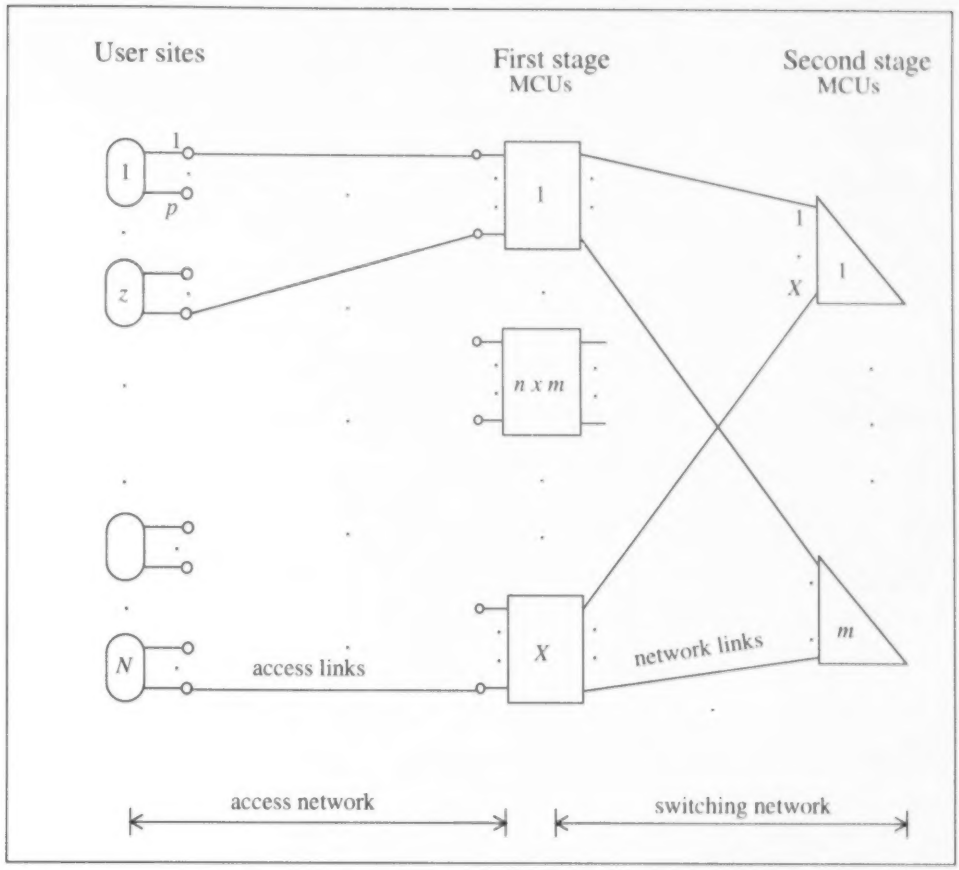


Figure 7.3. Network Topology

If z is the number of sites connecting to the same first stage MCU, then the number of ports required for customer access at each first stage MCU is:

$$n = p \cdot z \qquad \text{(Eq. 68)}$$

Given N sites in the network with p access ports each, and each first stage MCU having n ports for customer access, the number of first stage MCUs required is:

$$X = \left\lceil \frac{N \cdot p}{n} \right\rceil \qquad \text{(Eq. 69)}$$

If the total number of MCUs to be considered in the design is X_{max} and X is the number of MCUs required in the first stage, then the number of MCUs used in the second stage has to satisfy:

$$m \leq X_{max} - X \qquad \text{(Eq. 70)}$$

Substituting Equation 69 into the above equation we get:

$$m + \left\lceil \frac{N \cdot p}{n} \right\rceil \leq X_{max} \qquad \text{(Eq. 71)}$$

The variable m is constrained by Equations 67 and 70. For example, if $C=16$, $X_{max}=20$, $n=4$ and $X=10$, then according to Equation 67, the number of network links, m, is 12. But we know that the number of second stage MCUs, m, cannot be more than 10 according to Equation 70, because that would require a total of 22 MCUs in the design. Therefore, in this case not all C ports will be utilized at the first stage MCUs. On the other hand, if it is the number of network links that is the limiting factor, then all X_{max} of the MCUs may not be utilized. The number of second stage MCUs or the number of network links allowed on each first stage MCUs can therefore be calculated as:

$$m = min(C - n, X_{max} - X) \qquad \text{(Eq. 72)}$$

7.2.1 OPTIMIZATION PROCEDURE

The optimization theory is based on the dependency of end-to-end blocking on the allocation of the first stage MCU ports between customer access and inter-MCU junctoring. Site blocking and network blocking show opposite behaviour as the split is varied. Site congestion decreases as n increases whereas network blocking increases. Thus, there is an optimal split for which the sum of the blocking probabilities is a minimum. In order to calculate the minimum blocking, the value of n is varied for a particular network configuration of X_{max}, z, D and C. The value of n, which implies values of p, m and X, at which minimum blocking occurs is the optimal design for the assumed X_{max}. A complete network optimization is then done by reducing X_{max} as long as the optimized end-to-end

blocking remains below the target. The end-to-end blocking is calculated using the analytical model at a fixed value of the traffic originating from each site (A_o) in the network. Note that in varying n at each stage, the values of n are chosen so that p is a whole number.

7.3 TESTS & RESULTS

The optimization problem requires the following inputs: the number of sites in the network, N, and the number of sites connecting to the same first stage MCU, z, the maximum number of destinations allowed per multicast call, D, along with the distribution of the number of destinations per call, $P(d)$, the maximum number of ports on each MCU, C, and the originating offered traffic from each site, A_o. The complete optimization is then achieved by varying the value of X_{max}, the total number of MCUs considered while at each value of X_{max}, seeking the first-stage MCU port allocation that minimizes P_{end}.

To apply the optimization technique, a test network is used which has 20 sites with 2 sites connecting to the same first stage MCU. The maximum number of destinations per call was D=3, uniformly distributed, and each MCU had C=16 ports. The offered traffic intensity originating from each site in the network was assumed to be A_o= 0.1 erlangs. Using the optimization procedure, curves for the test network were generated with n varying as shown in Figure 7.4. For each value of X_{max}, the value of n was varied from 4 to 10 and the end-to-end blocking calculated analytically. As expected, there is an optimal value of n at which the end-to-end blocking is a minimum for each X_{max}. To illustrate the use of these optimization curves, assume that the service provider wants to meet the grade of service objective of 1.5% blocking at minimum cost for the test specifications listed above. Looking at the optimization curves, one solution is to choose a network configuration with n=6 and X_{max}=20, implying p=3, m=10 and X=10. But the same objective can be met with considerable cost savings by using a maximum of 16 MCUs if n is chosen to be 8, implying p=4, m=6 and X=10.

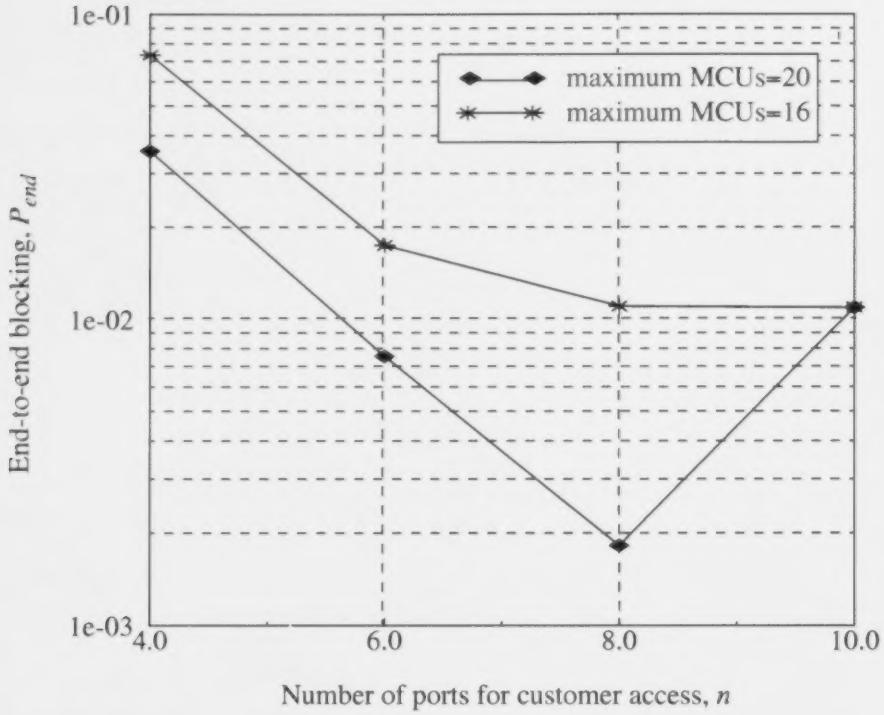


Figure 7.4. Optimization curves: *N=20, D=3, A_o=0.1 erl, z=2, C=16*

7.4 SUMMARY

An optimization method is presented for designing a multicast network to meet a desired blocking objective while using a minimum number of MCUs. The procedure involves varying the split of the MCU ports on each first stage MCU between customer access and inter-MCU junctoring. Site and network blocking show opposite behaviour as the split is varied: site blocking decreases as n increases whereas network blocking increases. Therefore the issue is to find the optimal split for which the end-to-end blocking in the network is a minimum for a fixed number of total MCUs and a fixed number of ports on each MCU. A complete optimization is done by reducing the maximum number of MCUs allowed as long as the optimized end-to-end blocking remains below the target.

8.0 VARIANTS ON THE NETWORK MODEL

So far in this thesis all of the test networks and performance evaluation methods have been based on the network model presented in Chapter 3. In this chapter, we consider some variations in the network model and consider their effect on the performance of the test networks.

8.1 OPTIMIZED MULTICAST ROUTING ALGORITHM IN EMULATOR

Each multicast call requests connections to d destination sites in the network and therefore makes d connection attempts. Connection attempts that are not site blocked or deflected are routed through the switching network one at a time using Dijkstra's shortest path algorithm. Therefore, the routing algorithm used in the emulator is greedy in the sense that it satisfies demand for a connection attempt through the switching network, regardless of its effect on the next connection attempt. In Figure 8.1, we give a scenario to illustrate how the sequential routing of connection attempts in a call is not strictly optimal in terms of blocking on a per call basis.

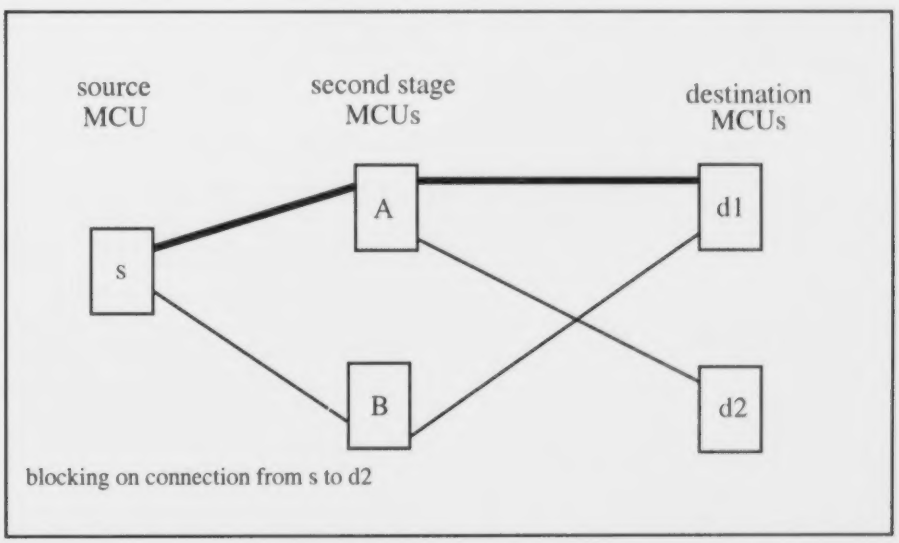


Figure 8.1. Greedy routing example (blocking on connection from s to d2)

Assume that source MCU *s* has to connect to destination MCUs *d1* and *d2* and that the connection attempt to *d1* is processed first. If all links shown are free, there are two routing options available to connect *s* and *d1*: *s-A-d1* and *s-B-d1*. Since the routing algorithm is greedy, it does not generate both options, but chooses the first option it finds. In the case where the chosen route is *s-A-d1*, the emulator will busy the links along that path and then go on to process the connection request to *d2* and will find then that there is no longer a path available to connect *s* to *d2*, because the only path available was through second stage MCU A, which was used to connect to *d1*. But going back to the connection request between *s* and *d1*, if the routing algorithm chooses *s-B-d1* as the first route, then it is possible to also satisfy the connection request to *d2* via MCU A (see Figure 8.2). Therefore, in this particular example, the greedy routing algorithm satisfies only one of the two requested attempts in a call even though both of them could have been satisfied.

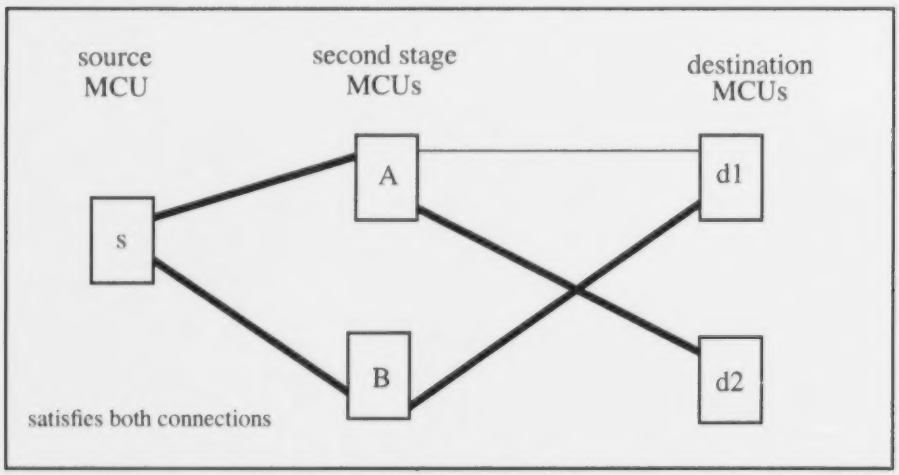


Figure 8.2. Optimal routing example

To maximize the number of successful connection attempts, we formulated the routing sub-problem in the emulator as an integer programming problem. Rather than processing each attempt for a call sequentially, the emulator makes a demand file that lists all of the destination sites. A destination site is only included in the demand if it is not site blocked and not deflected. The integer program then chooses a route set that maximises

the "flow", in this case number of successful connections. The successful connections are given back to the emulator. The emulator then processes each of the routes by setting the links busy, along with an access port at each successfully connected destination. The number of connection attempts that are not feasible are recorded as network blocked attempts.

To get an idea of the improvement in network performance achieved through this form of optimal multicast routing, we tested the new routing algorithm using design specifications for Network A described in Chapter 6. The results for the network using simple greedy routing were obtained previously and compared to the analytical model. Now we compare the blocking levels in Network A for optimal and greedy routing algorithms using the emulator.

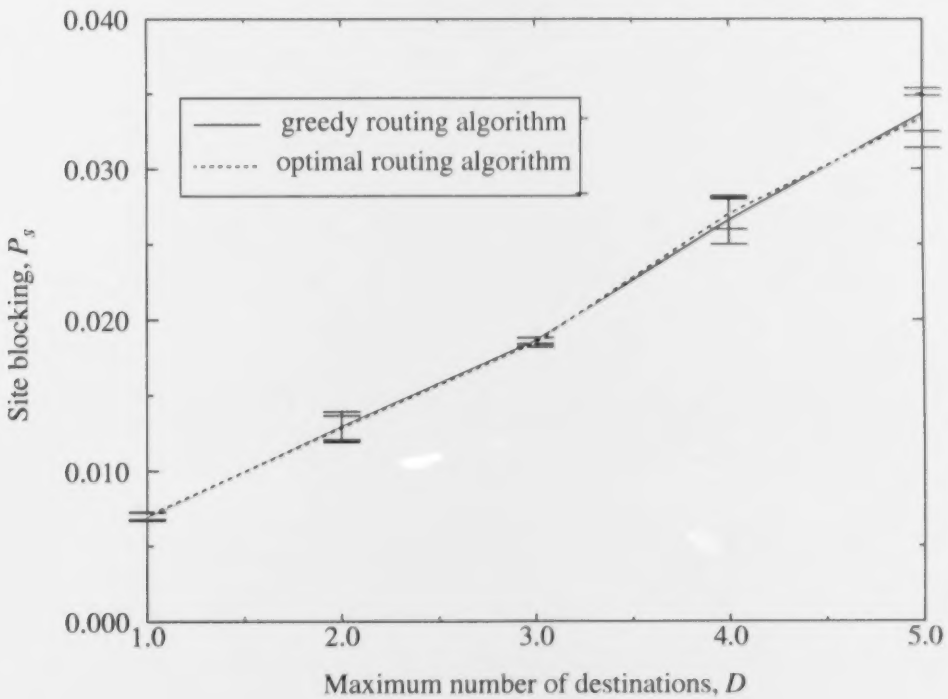


Figure 8.3. Site blocking in Network A: greedy vs. optimal routing

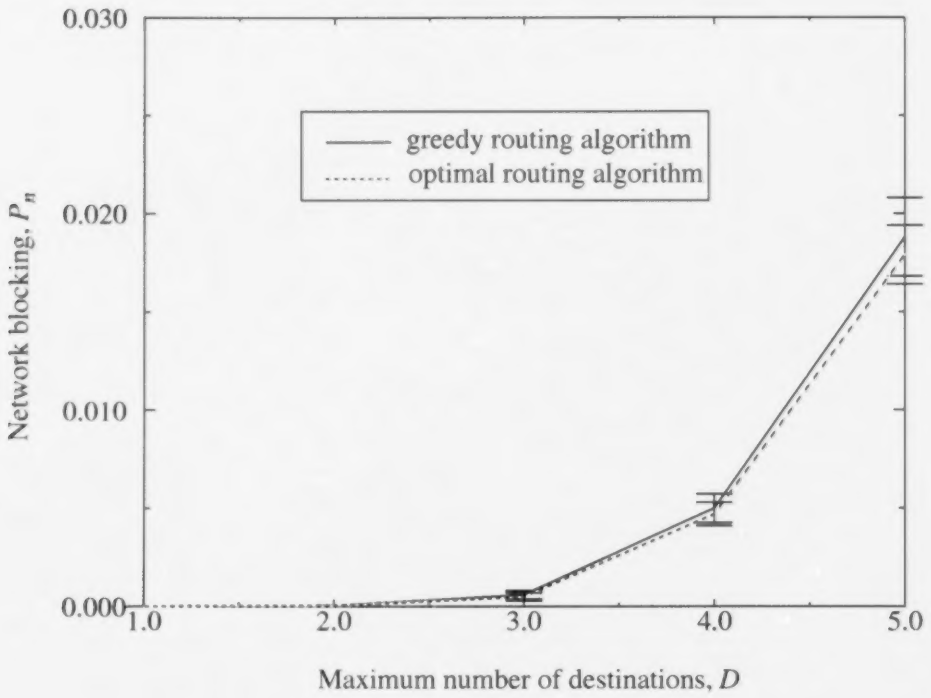


Figure 8.4. Network blocking in Network A: greedy vs. optimal routing

The results for site and network blocking (see Figures 8.3 and 8.4) show that the optimal routing algorithm gives very little improvement over the greedy one. In hindsight, the reason for this seems to be that the greedy algorithm leaves more idle network capacity which almost counteracts its own greediness. In addition blocking is low to start with so we are seeking to observe a second-order effect. If a route to a destination site is available, the bandwidth or the links required are occupied and "busied". Therefore, the routing algorithm is greedy on a per connection basis. As a result, attempts that have not been processed yet might get blocked, as illustrated in Figure 8.1. This leaves more free links in the network that can be utilized by future calls. For example, in Figure 8.1, if another call requesting a connection between *s* and *d1* arrived, it could be set up using the path *s-B-d1*. On the other hand, the optimal routing algorithm maximizes successful connection attempts on a per call basis. This algorithm is likely to use more network links on a per call basis and therefore block more attempts in the future. Looking at Figure 8.2, the optimal routing algorithm connects *s* to *d1* and *d2* and therefore another call arrival requesting

a connection between *s* and *dl* cannot be satisfied. Since the traffic is random in nature, the advantage of maximizing successful connections on a per call basis is minimal.

In practise therefore, neither of the algorithms is superior to the other in terms of blocking levels, but the greedy algorithm is easier to implement.

8.2 MULTIPLE INTER-STAGE LINKS

In the basic network model we make the assumption that there is exactly one link between each pair of MCUs in the first and the second stage of the switching network. As a result of this assumption, the number of network links per first stage MCU equals the number of second stage MCUs. A drawback of this assumption is that on certain occasions or for certain design configurations, all of the ports at the first stage MCUs may not be utilized. If the total number of available ports per MCU is *C*, there are (*C-n-m*) unused ports on each first stage MCU because of our assumption that exactly one link exists between each pair of first and second stage MCUs. The number of ports busy on each second stage MCU is equal to the number of MCUs in the first stage, *X*, which implies that there are *C-X* free ports available on each second stage MCU. In total there are $((C-n-m)\cdot X)$ free ports at the first stage and a total of $((C-X)\cdot m)$ free ports at the second stage. Thus, the unused ports on first stage MCUs could connect to the unused ports on the second stage MCUs. The heuristic for determining the multiple linking pattern could be based on demand or for specific customer circumstances.

To illustrate multiple inter-MCU links in a network, we refer to Network B as described in Chapter 6. There are 4 first stage MCUs, each having 6 ports for customer access and 8 ports for inter-MCU junctoring, and therefore the number of second stage MCUs is 8. If we assume that there are a total of 16 ports on each MCU, then there are 2 free ports per first stage MCU or 8 free ports in total at the first stage and 12 free ports at each second stage MCU for a total of 96 free ports at the second stage. To test our hypothesis that the network blocking will decrease with multiple links, we connect the free ports on each first stage MCU to the second stage sequentially, with each of the 8 free ports connecting to a second stage MCU. Therefore, each first stage MCU now has 10 network

links connecting to the second stage instead of the previous 8, with multiple links between pairs of first and second stage MCUs. The extra 88 free ports at the second stage can be used for future expansion of the service when more subscribers are added to the network. The emulator is used to test the performance of the new topology of Network B against the results presented in Chapter 6. As expected, the network blocking for the case with multiple inter-stage links between pairs of first and second stage MCUs was found to be lower (see Figure 8.5).

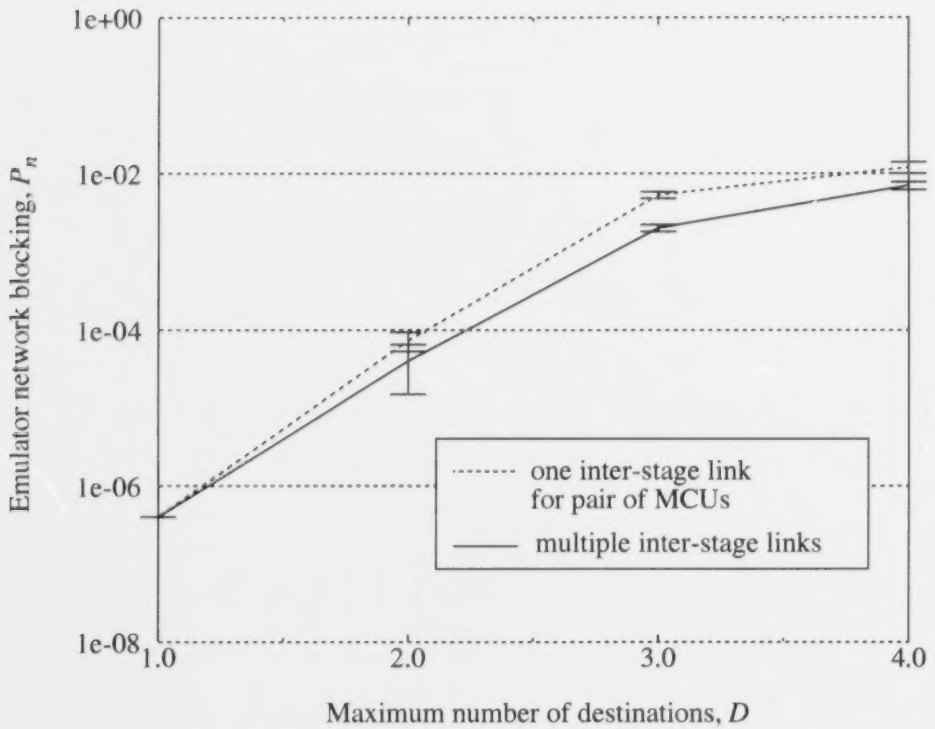


Figure 8.5. Network blocking in Network B: multiple inter-stage links

We do not presently attempt to modify the analytical model for network topologies having multiple links between some pairs of first and second stage MCUs due to the complications that arise in the multicast channel graph as a result of the multiple linking pattern. This brings to light one of the important practical differences in the nature of

operation of the emulator and the analytical model. Any changes in the network topology are easily handled in the emulator by changing the input network file which contains the network topology. But the analytical model may be intractably complex to handle any such non-uniformities in the network topology and is limited to symmetric network topologies. The emulator can be used to explore variations in network topology based on specific customer circumstances.

8.3 COMPLETE BLOCKING RATHER THAN PARTIAL CALL SETUP

All of the results presented so far in the thesis have been based on the assumption that all feasible portions of multicast calls will proceed even if some destinations are blocked. Thus, the event that is measured and calculated as multicast blocking is any unsuccessful connection attempt to connect a source-destination pair. This can be called "soft blocking". We now model a hard blocking, all or nothing assumption, in which a multicast call is set up if and only if all of the requested destination sites can be successfully connected to the source site.

Based on the hard blocking assumption, a multicast call will be said to be blocked unless all requested destinations can be connected to the source. If a call requests connections to d destinations and all d are successfully connected, the call is set up; otherwise it is said to be blocked. Any particular connection attempt could be blocked due to either destination or network blocking. If the source is site-blocked, that is also recorded as a blocked event. To draw a comparison between the performance of a network under soft and hard blocking assumptions, we make the desired changes in the emulator and test Network A for the hard blocking case and compare against the results obtained earlier using the soft blocking assumption.

The results shown in Figure 8.6 indicate that the hard blocking assumption introduces more end-to-end blocking in the network compared to the soft blocking assumption. This is due to the fact that in the all or nothing scenario, more network resources are used for satisfying a call with d connection requests, as opposed to the soft blocking case. The end-to-end blocking is the same for the case of point-to-point calls ($D=1$), because each

call request makes one connection attempt and therefore the probability of a whole call being blocked equals the probability of one connection attempt being blocked. As the maximum number of destinations allowed in a call increases, the discrepancy between hard and soft blocking increases.

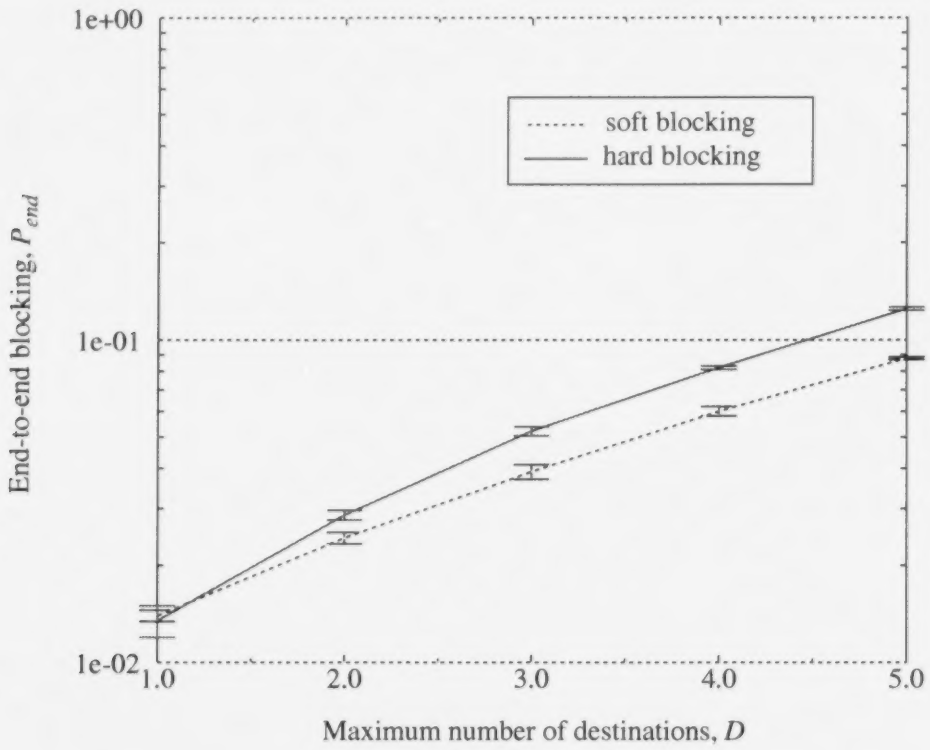


Figure 8.6. End-to-end blocking in Network A: soft blocking vs. hard blocking

8.4 SUMMARY

Some variations in the network model have been described and analysed using the emulator. The blocking levels in a test network using optimal routing for each call show no improvement over the greedy method. This is because the optimal method maximizes the number of successful connection attempts per call and therefore tends to block more connection attempts in future calls. Another variation in the network model was to make use of the free ports at the first and second stage MCUs after each pair of first and second stage

MCU have been connected using one network link. Thus certain pairs would have multiple network links between them. As expected, network blocking was lower when using multiple inter-stage links between some pairs of first and second stage MCUs as opposed to the case with only one link between the pairs. The third variant is the hard blocking assumption in which a multicast call is only set up if all d destinations can be successfully connected. The results for a test network under the soft and hard blocking assumptions show that end-to-end blocking in the hard blocking case is greater than for the soft blocking case.

9.0 SUMMARY, CONCLUSIONS AND FUTURE WORK

9.1 THESIS REVIEW

The central theme for this work was the performance and optimization of a particular class of multicast circuit-switched networks.

In Chapter 1 we introduced the need for better methods to optimize and evaluate the performance of multicast networks catering to the educational or business community. Recently, multicast control units (MCUs) have been developed that serve as the switching devices for multicast videoconferencing. The bridging and switching in these MCUs is relatively well understood, but it has been unclear how to optimally provision MCUs so that an overall blocking objective is met while using minimum resources.

Chapter 2 provided the relevant background to understand the work described in this thesis. In particular, the concepts of traffic, blocking and switching networks are detailed. These three topics form the basis of any telecommunication service and are extremely important in its design.

Chapter 3 presented the basic network model outlining the network topology and the assumptions about multicast calls, traffic and the routing. The network topology consists of access plus switching. The access component consists of the user sites and the access ports at each site. Switching provides the necessary interconnection between the users using a two-stage single-sided network of MCUs. The traffic was assumed to have exponentially distributed holding and inter-arrival times. Multicasting was achieved by assuming that the first stage MCUs have fanout capability. All feasible portions of multicast calls are processed. Connection attempts to destinations having access to the same first stage MCU as the source are deflected.

In Chapter 4 we documented the analytical model. This consists of a detailed derivation leading to a calculable model of end-to-end blocking in a multicast network based on the traffic scenario and the network dimensions. Calculating end-to-end blocking

involves aspects of Erlang theory for full-availability server groups and aspects of multi-stage switching network analysis via multicast channel graphs developed in this thesis.

Chapter 5 discussed the multicast network emulator which was designed to be a functional model of call-by-call handling in actual multicast networks. The emulator is also used to validate the analytical results and assumptions.

In Chapter 6 the analytical results for three test networks were compared to the emulation results and found to give good agreement. The performance metrics and key assumptions made in the analytical model were validated.

Chapter 7 developed the optimization theory used to minimize the number of MCUs required while maintaining the grade of service objective. This involves varying the partitioning of the first stage MCU ports between customer access and inter-MCU junctoring.

In Chapter 8 we tested variations in the network model using the emulator. In particular, we tested optimal routing and found that it shows little improvement over the greedy algorithm assumed earlier. Another variant tested was to have multiple network links between certain pairs of first and second stage MCUs. As expected, this variation reduces network blocking due to the availability of more parallel paths in the switching network.

9.2 CONCLUDING DISCUSSION

We presented an analytical model and an emulator to design and evaluate the performance of a class of multicast networks. The analytical model has been validated by discrete event simulation of the call arrivals and departures using a functional emulation of the actual network operation. We then used the analytical model to systematically develop optimum multicast networks with a minimum number of MCUs. The optimization technique allows network planners to meet a given blocking target by utilizing a minimum number of MCUs. The analytical model has been useful in providing insights about the fundamental factors influencing the performance of such networks but it cannot deal with an asymmet-

ric network design or irregular linking patterns or different numbers of access ports per site. The network emulator complements the basic design insights of the analytical approach by dealing with arbitrary case-by-case variations and may be useful in the practical design of multicast networks for specific customer circumstances.

9.3 FUTURE WORK

The work presented in this thesis is a good starting point for future analytical and simulation-based research in the area of multicast network design and optimization. There are a number of issues that have arisen out of this work.

The research done in this thesis focuses on two-stage single-sided switching networks. Variations of the network topology could be an area of further research. For example, for a large network, a three-stage network may be more suitable to minimize blocking. Through our work we have assumed that the multicast traffic is random in nature with exponentially distributed holding and inter-arrival times. An interesting extension of this work would be to introduce scheduling in the network design where the network would have advance knowledge of the demand it will be expected to carry in the future. In this case routing can be optimized to minimize blocking of such traffic. Another area of research is to extend the idea of the network model to desktop videoconferencing. In such a network, the Engset theory is applicable rather than Erlang B. This is due to the fact that if each site is a user in the network, then a busy user cannot generate another arrival. This introduces finite source effects. Another area of future work is to design an animation of the emulator, to graphically depict the network on a screen and show the connections being set up.

BIBLIOGRAHY

[1] Taylor, P.C., "Multipoint videoconferencing trends and techniques", *Proc. Olympus Utilisation Conference*, Vienna, April 1989, pp. 361-65.

[2] Halhed B, R., "MCUs: Technology for multipoint videoconferencing", *Business Communications Review*, Vol. 21. No. 5, May 1991, pp. 72-8.

[3] Oka, S., Misawa Y., "Multipoint teleconference architecture for CCITT standard videoconference terminals", *SPIE: Visual Communications and Image Processing*, 1992, pp. 1502-11.

[4] Clos, C., "A study of non-blocking switching networks", *Bell Syst. Tech. J.*, Vol. 32, March 1953, pp. 1481-92.

[5] Masson, G. M., "On rearrangeable and non-blocking switching networks", *Proc. ICC'76*, pp. (7-1)-(7-7).

[6] Benes, V. E., "On rearrangeable three-stage connecting networks", *Bell Syst. Tech. J.*, Vol. 41, Sept. 1962, pp. 1481-92.

[7] Masson, G. M. and Jordan, B. W., "Generalized multi-stage connection networks", *Networks* Vol 2, 1972, pp. 191-209.

[8] Hwang, F. K., "Rearrangeability of multi-connection three-stage clos network", *Networks* Vol 2, 1972, pp. 301-306.

[9] Yang, Y. and Masson, G., "Nonblocking broadcast switching networks", *IEEE Trans. Computers*, Vol. 40, No. 9, Sept. 1991, pp. 1005-15.

[10] Jajszczyk, A., "Comments on: Three-stage multi-connection networks which are nonblocking in the wide sense", *Bell Syst. Tech. J.*, Vol. 62, 1983, pp. 2113-14.

[11] Hwang, F. K. and Jajszczyk, A., "On nonblocking multiconnection networks", *IEEE Trans. Commun.*, Vol. 34, 1986, pp. 1038-41.

[12] Hills, M. T., *Telecommunications Switching Principles*, Allen & Unwin, London, 1979.

[13] Freeman, R. L., *Telecommunication System Engineering*, John Wiley & Sons, New York, 1980.

[14] Wang, W., Saadawi, T. N., "Trunk congestion control in heterogeneous circuit switched networks", *IEEE Trans. Commun.*, Vol. 40, No. 7, July 1992, pp. 1156-61.

[15] Labetoulle, J., "Mixing of traffic on a trunk group: Calculation of blocking probabilities using an extension of the ERT method", *ITC-11*, 1985, pp. 945-51.

[16] Liao, K. and Roberts, J., "Videoconferencing traffic and network design", *IEEE Trans. Commun.*, Vol. 35, No.3, March 1987, pp. 275-82.

[17] Roberts, J. and Liao, K., "Traffic models for telecommunication services with advanced capacity reservation", *ITC-11*, 1985, pp. 8-14.

[18] Zegura, E. "Evaluating blocking probability in generalized connectors", *IEEE/ ACM Trans. on Networking*, Vol. 3, No. 4, Aug. 1995, pp. 387-98.

[19] Chow, C., "On multicast path finding algorithms", *Proc. IEEE Infocom'1991*, pp. 1274-83.

[20] Leung Y., and Yum, T., "Efficient algorithms for multiple destinations routing", *Proc. ICC'91*, pp. 1311-17.

[21] Harrison T. G., *Performance Evaluation of Routing Algorithms for Multicast Traffic*, MSc. Thesis, University of Saskatchewan, Aug. 1995.

[22] Lee, K., Gersht, A. and Friedman A., "Multipoint connection routing", *Int. J. Dig. and Analog Comm. Sys.*, Vol. 3, 1990, pp. 177-86.

[23] Grover W. D., *Course notes for EE 589: Telecommunications networks and Systems*, University of Alberta, Edmonton, 1994.

[24] Jajszczyk, A., "One-sided switching networks composed of switching matrices", *IEEE Trans. Commun.*, Vol. 35, No. 12, Dec. 1987, pp. 1383-84.

[25] MacGregor, M, H., *The Self Traffic-Engineering Network*, Ph.D Dissertation, University of Alberta, Edmonton, 1991.

[26] Dijkstra, E. W, "A note on two problems in connection with graphs", Numer. Math. Vol. 1, 1959, pp. 269-71.

APPENDIX A

Algorithm for calculating the state probabilities of network links

OBJECTIVE: Calculate the probability of k network links being busy out of m, given d destinations per multicast call, i.e find $P_b(k)$.

GIVEN:

1) The probability distribution of the access links without deflection i.e., the probability of *i* access links being busy out of *n* for *d* destinations per call.

$$P_{ac}(i) = \sum_{i=0}^{n} \binom{n}{i} \cdot \left(a'_c\right) \cdot \left(1 - a'_c\right)$$

where $a'_c = a_c \cdot (1 - P_{def})$

2) Probabilities of an access link being in use connected to a source or a destination, p_s and p_d respectively.

MAPPING ALGORITHM:

IF *(k=0)* **{**

$$P_b(0) = P_{ac}(0)$$

}

IF $(1 \le k \le d-1)$ **{**

$$P_b(k) = P_{ac}(k) \cdot (p_d)^k$$

IF $(k=d-1)\{$

$$\chi_{d-1} = P_{ac}(1) \cdot (p_s) \cdot [P_s + P_n]$$

$$P_b(k) = P_b(k) + \chi_{d-1}$$

$\}$

$\}$

IF $(k=d)\{$

$$\chi_d = P_{ac}(2) \cdot \binom{2}{1} \cdot (p_s) \cdot (p_d) \cdot [P_s(d) + P_n(d)]$$

$$\Phi = 2, \kappa_1 = 1, \tau_1 = 0, \kappa_2 = 0 \text{ and } \tau_1 = d$$

$$P_b(d) = \left\{ \sum_{i=1}^{\Phi} P_{ac}(\kappa_i + \tau_i) \cdot (p_s)^{\kappa_i} \cdot (p_d)^{\tau_i} \right\} + \chi_d$$

$\}$

IF $(d+1 \leq k \leq m)\{$

$$\xi = \frac{k}{d}$$

IF(ξ *is a whole number*)$\{$

$$\Phi = \xi + 1$$

$\}$ *else* $\{$

$$\Phi = \lceil \xi \rceil$$

$\}$

$$\kappa_i = \Phi - i\,,\ \tau_i = k - (\kappa_i \cdot d)$$

$$P_b(k) = \sum_{i=1}^{\Phi} \binom{\kappa_i + \tau_i}{\kappa_i} \cdot P_{ac}(\kappa_i + \tau_i) \cdot (p_s)^{\kappa_i} \cdot (p_d)^{\tau_i}$$

IF(($k = \Lambda \cdot d$) *and* Λ *is a whole number*>*1*){

$$\chi_{(\Upsilon \cdot d)} = P_{ac}(\xi + 1) \cdot \binom{\xi + 1}{1} \cdot (p_s)^{\xi} \cdot p_d \cdot [P_s + P_n]$$

$$P_b(k) = P_b(k) + \chi_{(\Upsilon \cdot d)}$$

}

IF (($k = (\Lambda \cdot d) - 1$) *such that* Λ *is a whole number* >*1*){

$$\chi_{(\Lambda \cdot d) - 1} = P_{ac}(d, \Lambda) \cdot (p_s)^{\Lambda} \cdot [P_s + P_n]$$

$$P_b(k) = P_b(k) + \chi_{(\Lambda \cdot d) - 1}$$

}

}